НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система розподілу завдань між розробниками»

Виконав:
студент IV курсу, групи ІО-63
Косенков Ігор Володимирович _____

Керівник:
старший викладач
Алещенко Олексій Вадимович _____

Консультант нормоконтроль:
проф. д. т. н.
Сімоненко Валерій Павлович _____

Рецензент:
Радченко

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ СЕРГІЙ СТІРЕНКО
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Косенкову Ігорю Володимировичу

1. Тема проєкту «Система розподілу завдань між розробниками», керівник проєкту Алещенко Олексій Вадимович, старший викладач, затверджені наказом по університету від «07» травня 2020 р. №1081-С

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту *Технічна документація. Прикладний програмний інтерфейс. База даних MySQL. Середовище розробки IntelliJ IDEA, Java*

4. Зміст пояснювальної записки *Аналіз предметної області, дослідження методики побудови клієнт-серверних систем на принципі MVC, розробка системи розподілу завдань між розробниками.*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) функціональна схема, схема взаємодії, структурна схема.

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	*Затвердження теми роботи*	*01.09.2019*	
2	*Вивчення та аналіз завдання*	*30.03.2020*	
3	*Розробка архітектури та загальної структури систем*	*10.04.2020*	
4	*Розробка структур окремих підсистем*	*22.04.2020*	
5	*Програмна реалізація системи*	*04.05.2020*	
6	*Оформлення пояснювальної записки*	*17.05.2020*	
7	*Передзахист*	*26.05.2020*	
8	*Захист*	*16.06.2020*	

Студент Ігор КОСЕНКОВ

Керівник Олексій АЛЕЩЕНКО

Анотація

У даній бакалаврській роботі описана система розподілу завдань між розробниками. У сучасному світі, коли над одним проектом працюють декілька людей, виникає питання, як оптимізувати їх роботу. Адже нам не потрібно, щоб працівники виконували одні й ті ж самі завдання й витрачали на це час. Для цього були розроблені спеціальні додатки та сайти, в яких ми можемо одночасно бачити які завдання треба виконати, які вже кимось виконуються, а які вже виконались.

У практичній частині розроблено прикладний програмний інтерфейс, за допомогою якого можна створювати завдання, які прив’язані до проекту. Завдання призначаються розробнику, в якій він записує скільки часу витратив. Також у завдань змінюються статуси, по яким ми бачимо на якому етапі розробки вони знаходяться.

Annotation

This bachelor's work describes a system for distributing tasks between developers. In the modern world, when several people work on one project, the question arises of how to optimize their work. After all, we do not need workers to perform the same tasks and spend time on it. For this, special applications have been developed in which we can at the same time see the tasks that need to be performed, which are already being carried out by someone, and which have already been completed.

In the practical part, an application programming interface has been developed with which you can create tasks that are tied to a project. Tasks are assigned to the developer, in which he writes how much time he spent on it. Also, in the tasks the statuses change, according to which we see at what stage of development they are.

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система розподілу завдань між розробниками”

Київ – 2020 року

Технічне завдання до *дипломної* роботи

Зміст

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Система розподілу завдань між розробниками».

Область застосування: різноманітні проекти на яких задіяні команди будь-якого розміру.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський Політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка системи розподілу завдань між розробниками, за допомогою якої, робітники зможуть відстежити актуальну інформацію по проекту: які завдання треба взяти у роботу, які вже виконують колеги, а які вже виконані.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування, бакалаврські роботи інших студентів, публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

- Створення проекту, додавання до нього членів команди, тим самим даючи доступ до нього.
- Створення завдань, прив’язаних до проекту, призначення працівника на нього.
- Створ1ення аккаунту працівника.
- Можливість записувати на яке завдання, скільки часу витратили працівники.

5.2. Вимоги до програмного забезпечення

- Операційна система Linux, macOS, Windows або Android чи iOS для смартфонів;
- Доступ до мережі Інтернет;
- Java Runtime Environment.

5.3. Вимоги до апаратної частини

- Комп’ютер на базі процесора Intel або AMD мінімальної потужності;
- Оперативної пам’яті не менше 512 Мбайт;
- Вільне місце на жорсткому диску не менше 256 Мбайт.

6. **ЕТАПИ РОЗРОБКИ**

	Дата
Вивчення літератури	30.01.2020
Складання і узгодження технічного завдання	11.02.2020
Створення модулів системи, що розробляється	14.03.2020
Тестування окремих модулів системи	10.04.2020
Доопрацювання, налагодження і виправлення помилок	04.05.2020
Оформлення документації дипломної роботи	17.05.2020

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

до дипломної роботи
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “Система розподілу завдань між розробниками”

Київ – 2020 року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	ДП.467100.001 ТЗ	Технічне завдання	4	
3	А4	ДП.467100.002 ВП	Відомість проєкту	1	
4	А4	ДП.467100.003 ПЗ	Пояснювальна записка	68	
5	А3	ДП.467100.004 Д1	Схема структурна Діаграма класів	1	
6	А3	ДП.467100.005 Д2	Схема функціональна Блок-схема алгоритму	1	
7	А3	ДП.467100.006 Д3	Схема взаємодії	1	

					ІАЛЦ.467100.002 ВП
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	
Розроб.		*Косенков І.В.*			*Відомість дипломного проекту*
Перевір.		*Алещенко О.В.*			
Н. Контр.		*Сімоненко В.П.*			
Затверд.					

Літ.	*Арк.*	*Акрушів*
	2	*1*

НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІО-63

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту

на тему: “Система розподілу завдань між розробниками”

Київ – 2020 року

ЗМІСТ

ВВЕДЕННЯ

На попередніх курсах в університеті у нас були групові завдання. Основна мета таких завдань навчити студентів роботі в команді. Обирається головний у команді, який повинен розподілити завдання між своїми одногрупниками. Постає питання: «як це зробити?». Можна зібратися після занять в університеті і розповісти кожному його завдання у всіх подробицях. Але ці настанови від лідера команди можуть швидко забутися вже у ввечері цього дня. Інший варіант – це записати кожне завдання у текстовий файл і завантажити кожному на USB-флеш-накопичувач або жорсткий диск. Але не для кого це не секрет, що під час розробки часто виникають зміни у поставлених завданнях, бо до кожної дрібниці все наперед неможливо продумати. І за таких умовин цей спосіб розподілу завдань нам не підходить. Щоб швидше вносити зміни до завдань і надсилати їх до колег, треба робити це онлайн. Тобто той текстовий документ із завданнями ми можемо відправити всім учасникам команди по елетронній пошті, соціальним мережам та месенджерам. Це вже ефективніше і зручніше попередніх двух способів, але все одно лідеру команди потрібно кожен раз відправляти завдання кожному адресату окремо. А розробники можуть їх загубити серед особистих повідомлень, рекламних розсилок і т.д. Отже, нам потрібна окрема онлайн система, в якій лідер команди зможе створити завдання, або змінити існуюче в одному місці, і це буде видно всім представникам команди.

Після аналізу цих вхідних даних було прийнято рішення зробити систему розподілу завдань між розробниками. Але ця система может бути застосована у проектах з різних сфер діяльності.

Система складається з серверної частини та графічного інтерфейсу користувача. В рамках бакалаврської роботи розроблена серверна частина, яку вже можна інтегрувати з будь-яким графічним інтерфейсом, наприклад веб-сайт, мобільний додаток, чи десктопне рішення.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

1.1. Опис завдання

Завдання даної роботи полягає у створенні прикладного програмного інтерфейсу для систем розподілу завдань між розробниками, яке дозволить користувачам створювати завдання, змінювати їх і назначати на них працівників.

Система повинна розрізняти ролі: адміністратор і звичайний користувач. Адміністратору потрібно надати права на створення проекту і надання доступу працівників до нього. Звичайний користувач може створювати завдання, змінювати його, призначати на них себе чи інших працівників.

Додатковими функціями системи є записування часу, який працівник витратив на певне завдання, а також розширені параметри завдання, такі як статус, преорітет, оцінка потрібного часу на завдання і т.д. Все це допомагає розширити базовий функціонал розподілу завдань, підлаштувавши систему до методологій розробки таких як Agile, Scrum, Kanban.

1.2. Аналіз існуючих рішень

Додаток/сайт з управління проектами повинен включати в себе як мінімум такі завдання:

- призначення відповідальних осіб за проект/завдання;
- створення нових і редагування старих завдань;
- постановка дедлайнів;
- контролювання процесу роботи;
- звіт за часом роботи.

Отже, така система управління дуже корисна для різних компаній, щоб керівники могли контролювати своїх співробітників, бачити прогрес виконання роботи, дотримання всіх термінів, а працівник може побачити повний обсяг роботи і дедлайни.

Trello

Trello - це сайт, який виглядає як робочий стіл, де видно дошки. Їх можна створювати, редагувати, створювати списки і картки. Сайт успадковує принципи роботи системи канбан, що відрізняється від інших to-do листів, тому що можна бачити всі завдання на одній дошці. Trello розроблений на базі MongoDB, Node.js і Backbone.js. [1]

Канбан - один з принципів управління проектом. Суть є в тому, що на дошці розташовані картки в кілька рядів, що означає - етапи роботи над проектом. В процесі роботи проекту ці картки можуть переміщатися з одного ряду в інший, тобто кожен етап проходить цикл і доходить до завершального етапу - готово. Так візуально можна контролювати розподіл обов'язків на кожному етапі і дотримання рівномірності.

Основний вигляд сайту позначається на рис.1.1.

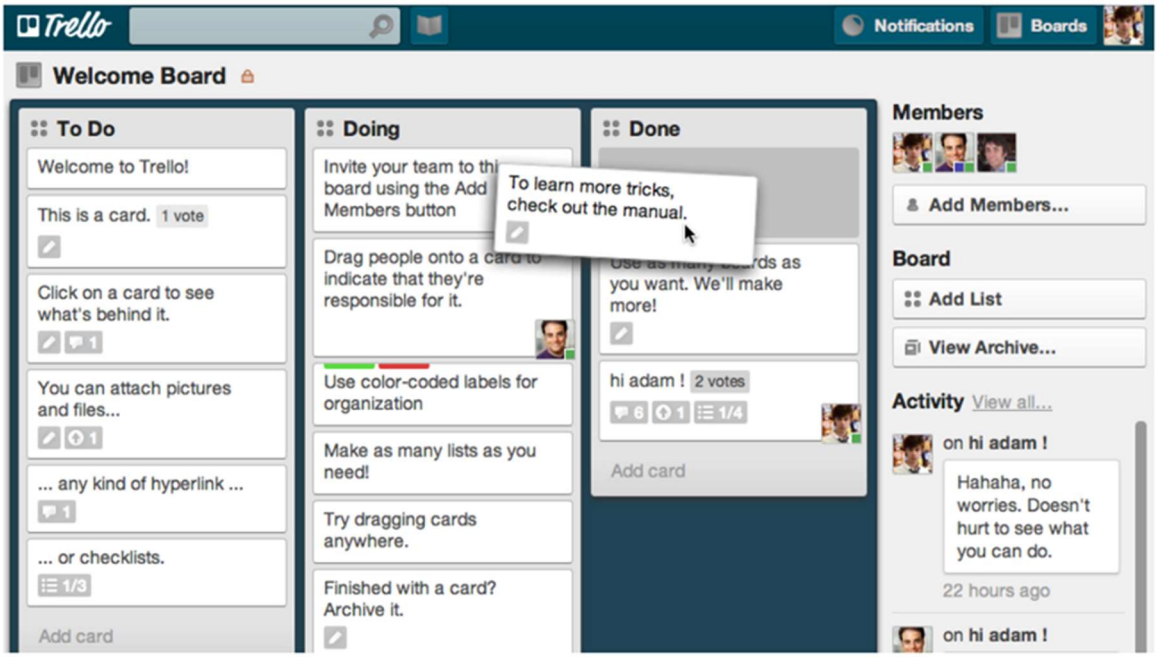


Рис.1.1. Загальний вигляд застосунку Trello [1]

Так як трелло - дошка з картками, то зазвичай там три ряди карток, які складаються з:

- що потрібно виконати;
- що робиться на даний момент;
- готово.

Ці ряди йдуть у вертикальному порядку. На картках вказуються завдання і описи до них. Завдання можна пересувати з одного списку в інший, або переміщати в різні дошки. Переміщення списків теж можливо. До будь-якого завдання можна призначити відповідальних осіб і тих, хто виконує завдання. Сервіс дає багато можливостей, включаючи шаблони для оформлення, певні настройки і управління елементами. Там можна обрати вже готові шаблони робочого столу під різний вид діяльності, будь це бізнес-проект чи щоденник. [2]

На картках в трелло можна виконувати такі функції:

- перейменовувати, заповнювати опис до завдання і редагувати текст, робити нотатки;
- привласнювати мітки (наприклад, ступінь важливості завдання), учасників для виконання, термін виконання, можна додати потрібні файли, є можливість додавання чек-листа;
- додавати коментарі, вставляти файли, документи, сповіщати учасників;
- змінювати положення завдання в списку, переміщати його за списками по іншим дошкам;
- копіювати, дивитися за змінами, архівувати;
- роздруковувати, ділитися посиланням на картку. [2]

Основні переваги Trello:

1. Сервіс пропонує стежити як за великими, проектами, так і за маленькими, можна додавати безліч список і карток.

2. Є безліч потрібних інструментів: дошка, картка, списки, контроль за проектом можливий в найдрібніших деталях. Одна картка може мати дві функції:

- одна картка = одна задача;
- одна картка = цілий проект, де можна додавати завдання у вигляді чек-листа, документи і різні коментарі.

3. Кожна задача може мати різного виконавця.

4. Trello - сервіс, який можна використовувати як для компаній, щоб регулювати процеси, так і для особистого користування у вигляді щоденника.

5. Завдання можна переміщати в інший список, вгору або вниз для того, щоб було легше розрахувати час.

6. Є різнокольорові мітки. З їх допомогою можна виділити якийсь етап, якщо виникла проблема або питання, або ж можна таким способом розставити акцент і пріоритетність завдання. [1]

Недоліки:

1. Не зовсім адаптована під невеликі пристрої. Не завжди зручно працювати, наприклад з телефону.

2. Працює тільки в онлайн режимі.

3. Наявність тільки мобільних версій.

4. Є платною для розширених можливостей.

Jira

Jira - система для управління проектами, призначена для стеження за помилками. Вона призначена для тих проектів, які мають відношення до Agile Scrum, Agile Kanban. [3]

Якщо бути точніше, це онлайн сервіс для управління проектами, є сайт, а також мобільний додаток. Система заснована на методології Agile, яка виглядає як дошка з картками, система схожа на Trello. [5]

Завданням можна надавати назви, певну тему, пріоритетність і коментарі до завдань. На завдання можуть бути призначені співробітники, є можливість внести файли, наприклад фотографії, а також коментарі до нього. Кожне завдання має свій статус, "в роботі", "закінчений", статус можна запросто міняти і переносити у вкладку "готово". Є журнал, який записує будь-які зміни, які робляться на дошці.

Jira має багато можливостей для налаштувань: кожна програма може мати певний тип завдання з власним workflow, статусом. Крім цього, за допомогою схем можна налаштувати для кожного проекту особисті права доступу. Jira є

досить універсальним інструментом, є безліч інших завдань, таких як управління ризиками або управління вимогами. [3]

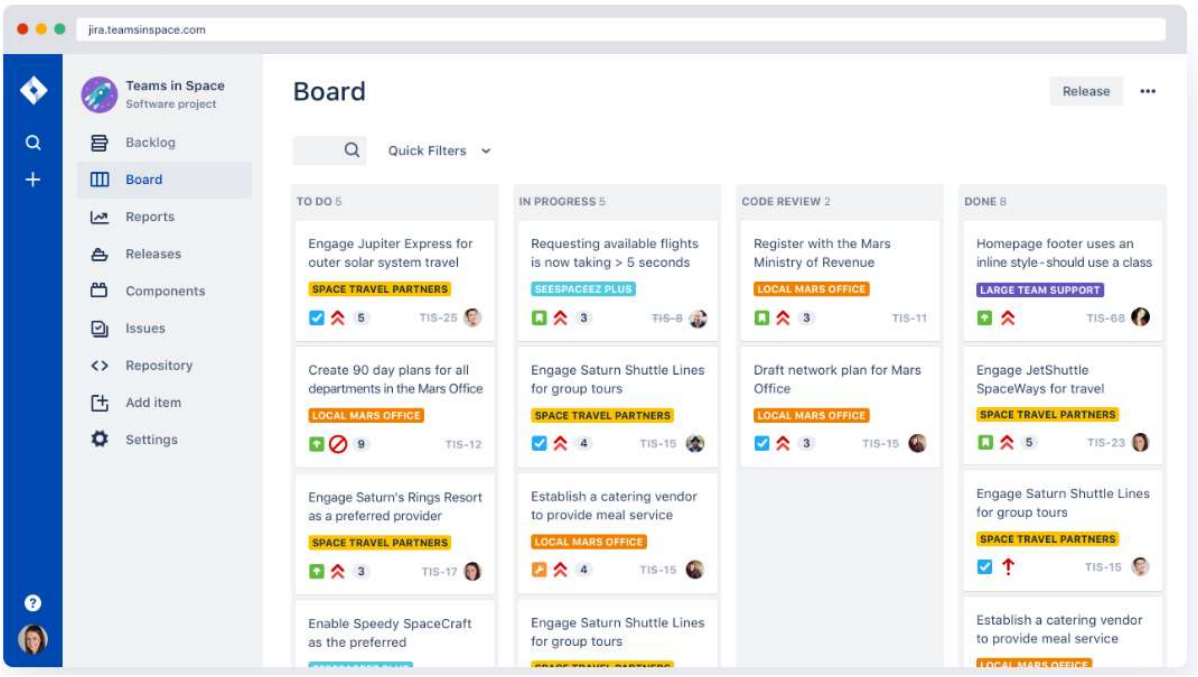


Рис.1.2. Загальний вигляд системи Jira [4]

Новий співробітник легко може додатися до будь-якого етапу проекту. Є можливість спільного редагування файлів, завдань.

Можна створювати окремі робочі групи, де кожен може побачити прогрес роботи і відслідковувати оновлення.

Jira дозволяє бачити весь процес повністю, заздалегідь помічати проблеми, помилки, для подальшого швидкого усунення. [6]

Основні переваги:

1. Простота в інтерфейсі та в користуванні.
2. Є канбан та scrum-дошки.
3. Можливість створення звітів.
4. Сервіс має інструменти, які дозволяють користувачам накреслити велику картину, повідомити плани та з'єднати більші цілі карти з повсякденними завданнями команди.
5. Програмне забезпечення забезпечує єдиний перегляд для всіх історій користувачів та може генерувати необхідні звіти для різних спринтів, таких як

діаграми виходу, швидкість спринту та інші. Також користувачі можуть стежити за навантаженням та завданнями команди.

6. Доступна інтеграція. Програмне забезпечення для відстеження випусків та проектів інтегрується з багатьма популярними сторонніми програмами. Наприклад, інтеграція з Hipchat і Slack полегшує спілкування з проблемами та реагування на сповіщення.

7. Користувачі можуть створювати та налаштовувати різні елементи, такі як таблиці, форми, терміни, звіти, тощо.

8. Працює для різних типів користувачів. Програмне забезпечення може використовуватися розробниками, керівниками проектів, інженерами, менеджерами та іншими нетехнологічними бізнес-професіоналами. [7]

9. Можливість побудови діаграми Ганта (чого немає наприклад у Trello).

Недоліки:

1. Складний мобільний додаток.

2. Звіти не можна використовувати повторно. Графічні звіти Jira не можна завантажити як зображення.

3. Обмежений розмір файлу. Користувачі скаржаться на те, що розмір файлу розміром файлів обмежений 10 МБ або менше. Тому завантажувати фото, відео чи документ розміром більше 10 Мб не дозволяється.

4. Заплутаний інтерфейс користувача. Деякі люди вважають, що інструменти фільтрації не найпростіші у використанні. Також деякі вважають конфігурації заплутаними, оскільки деякі частини програмного забезпечення все ще використовують більш стару версію свого графічного інтерфейсу, а деякі - новіші. [7]

Так як Jira та Trello відносяться до одного виробника (Atlassian), доречно зробити порівняльну характеристику цих платформ (табл.1.1).

Таблиця 1.1 - Порівняльна характеристика Jira та Trello

	Jira	Trello
Співпраця в команді	Jira пропонує своїм користувачам широкий спектр функцій співпраці всередині питань, таких як обмін файлами, обмін миттєвими повідомленнями, призначення завдань, члена команди та сповіщення. Сервіс пропонує більшу кількість функцій, які корисні для керівників проектів, які хотіли б організувати свою роботу в спринтах. Jira була побудована спеціально для забезпечення розробки програмного забезпечення, і вона підтримує Scrum, Kanban та змішані методики.	Можна запросити будь-яку кількість людей. Це означає, що всі користувачі отримають однакову видимість статусу та завдань проекту. Щоб спілкуватися, користувачі можуть коментувати картки, позначати інших товаришів по команді, ділитися вкладеннями. Можна призначати користувачів до певних карток.
Управління проектами	Поточні робочі процеси призначені для того, щоб допомогти командам створювати програмне забезпечення, але можна створювати і власні робочі процеси, що охоплюють інші сфери, такі як розробка продукту. Функції управління Jira пропонують велику гнучкість у налаштуванні процесу, а також можна створити дошку для команди. Як і в Trello, можна створювати завдання у вигляді питань, де додається вся необхідна інформація та призначається певним користувачам.	Можна легко організувати роботу за допомогою ієрархії дошок, списків та карток. Можна створювати та упорядковувати ці елементи за бажанням, призначаючи картки користувачам, додаючи терміни і додаючи файли. Дошка Trello - це дошка Kanban, поділена на стовпці, які відображають етапи робочого процесу та заповнені картки, де розміщується важлива інформація про завдання, доступна для всієї команди для підвищення її прозорості та продуктивності.

Продовження табл.1.1

Ціни	Jira не пропонує безкоштовний варіант. Jira поставляється в декількох різних версіях, таких як Jira Software, Jira Core та Jira Service Desk.	Основна версія Trello доступна безкоштовно. Безкоштовна версія Trello дозволяє користувачам отримувати переваги від усіх її функцій управління проектами і не встановлює обмежень на кількість дощок, які можна створити. Також Trello пропонує платні версії: Бізнес-клас та Enterprise. Ці версії дозволяють синхронізувати дані з іншими бізнес-системами, які використовуються командою, такими як Salesforce, Slack або Github.
Інтеграції	Jira перевершує Trello своєю масовою бібліотекою вбудованих інтеграцій за всіма видами рішень для відстеження часу, управління тестами, звітування, електронна пошта, огляд коду та хмарне зберігання. Jira пропонує безкоштовний API для розробників для клієнтів, які також хочуть налаштувати інтеграцію вручну.	Trello пропонує список власних додатків та інтеграційних підключень, включаючи Zendesk, Slack, Github, Salesforce та Google Drive.

Redmine

Redmine - ще один веб майданчик для управління проектами та відстежування помилок. Інтерфейс за відгуками користувачів є інтуїтивним, але все одно в ньому легко розібратися і можна швидко освоїтися. Сайт має безліч можливостей, хоч і є досить простим інструментом. Основні функції:

- можливість відслідковування помилок;
- налаштування статусів завдань;
- відстеження скільки йде часу на завдання, проект, а також контроль за співробітниками;

- облік часу на виконання завдання, є діаграма Ганта і календар;
- зміст для кожного проекту (Wiki);
- можливість додавати файли і документи;
- багатомовний інтерфейс;
- можливість розподіляти ролі між співробітниками і надавати їм певний доступ;
- нові користувачі можуть самостійно зареєструватися;
- є додаткові плагіни, для розширення функціональності веб-ресурсу. [9]

Стеження за завданнями і підписками на завдання встановлено певними механізмами. За кожним завданням можна вказати спостерігача, після чого, коли будуть встановлені якісь завдання або додані файли, коментарі, співробітники отримають оповіщення/повідомлення на електронну пошту.

У Redmine передбачена діаграма Ганта, що у багатьох сайтів такої функції немає. Більш того, встановивши додаткові плагіни, можна легко формувати звіти, щоб бачити статистику і на якому етапі знаходиться проект.

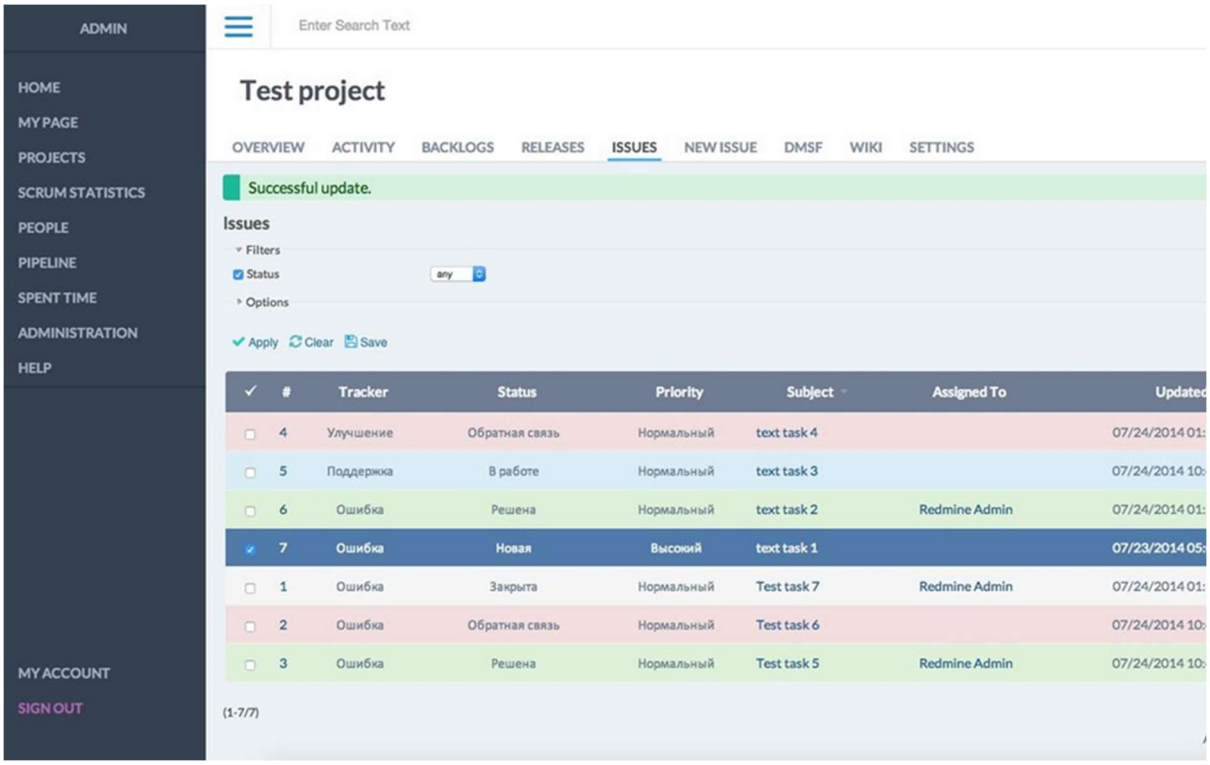


Рис.1.3. Загальний вигляд сервісу Redmine

Переваги:

1. Відстежування прогресу проектів та відповідальності членів команди.
2. Контроль таких показників як час, витрачений командою або певними особами, є функція створення задач та слідкування за окремими функціями проекту.
3. Є можливість отримувати звіти про час, витрачений на кожну категорію в проекті, і це дозволяє приймати більш обґрунтовані рішення для бізнесу.
4. Він має багато функцій, які необхідні для управління проектами, таких як діаграми Ганта, призначення завдань і розміщення локально. Користувачі можуть призначати завдання і терміни.
5. Простота інтерфейсу.
6. Гарний інструмент як для повсякденних завдань так і для бізнес-рішень.
7. Є плагіни, які можна використати для розширення можливостей в роботі.
8. Його API дозволяє легко інтегруватися з іншими програмними забезпеченнями.
9. Безкоштовність.

Недоліки:

1. Можливість налаштування і можливість взаємодії із зовнішніми додатками можуть зробити веб-ресурс нестабільним через недотримання робочого процесу.
2. Дизайн користувальницького інтерфейсу повинен бути поліпшений, оскільки іноді функціональні можливості інструменту не є інтуїтивно зрозумілими, і це призводить до втрати часу. Крім цього, інтерфейс виглядає застарілим.
3. Важкий в налаштуваннях.
4. Не так багато функцій та можливостей, як у багатьох конкурентів.

Jira є найбільш популярним сервісом для управління проектами серед користувачів. Тому для Redmine – це головний конкурент.

В загальному Jira в основному зосереджена на конкретному дизайні, щоб допомогти користувачам виконувати свої завдання та присвоювати їх окремому користувачеві на основі пріоритету. Програма є платною, коштує від 7$. Є різні підписки – річна, щотижнева чи одноразова. Загальний бал – 9,3 (по опитуванням користувачів). Redmine – гнучкий сервіс для декількох бізнес-цілей. Він написаний на мові Ruby на основі фреймворку. Redmine завжди був незалежним від платформи і бази даних. Платформа має безкоштовний доступ. Але його загальна оцінка – 8 з 10, в той час як у Jira – 9,3. В нього дуже мала функціональність. [11]

Features	JIRA	RedmineUP Cloud
Issue tracking	✓	✓
Time tracking	✓	✓
Scrum/ Kanban boards	✓	✓
Agile charts, such as Burndown	✓	✓
Gantt	[Requires Add-on]	✓
Helpdesk	[Requires Add-on]	✓
FAQ, Q&A, Knowledge base	[Requires Add-on]	✓
Built-in source control integrated WIKI	[Requires Add-on]	✓
Printable PDF reports	[Requires Add-on]	✓
Git/SVN/CSV integration	[Requires Add-on]	✓
Employee and team management	[Requires Add-on]	✓
Contacts, companies, deals		✓
Client access		✓
User rates, Invoicing, Accounting		✓

Рис.1.4. Порівняльна характеристика функцій Jira та Redmine [10]

Asana

Asana - це додаток для проектного менеджменту, використовується в індивідуальних цілях або для різних розмірів проектів компаній. Включає в себе безліч функцій. Основний акцент програми - це те що керувати проектом можна і без повідомлень по електронній пошті. Інтерфейс ресурсу досить простий, дуже схожий на інші ресурси з управління проектами.

Кожен проект може включати в себе команду, де кожен створює для себе робоче середовище. На робочому столі в додатку може розміщуватися не один проект, а кілька, що в свою чергу включає багато завдань. Співробітники можуть прикріплювати документацію, додаткові файли, коментувати завдання, ставити мітки, для цього їм потрібен певний доступ до редагування.

У разі зміни завдання, або додавання коментарів, співробітник моментально отримує сповіщення. [12]

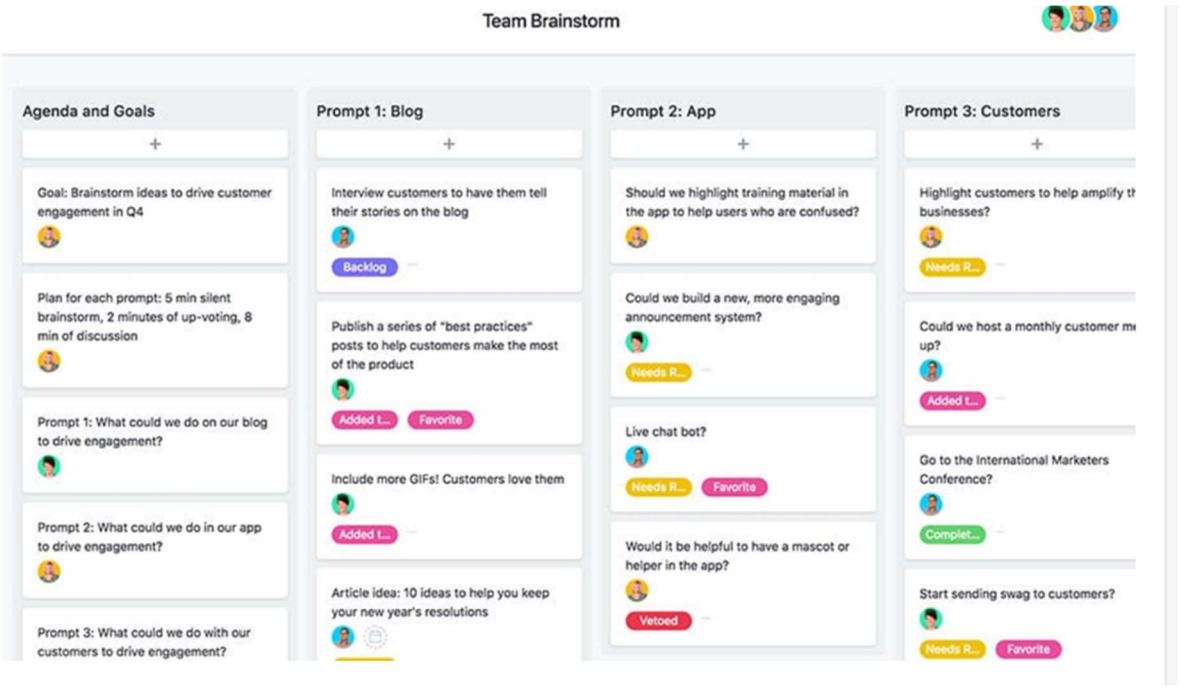


Рис.1.5. Загальний вигляд платформи Asana

Користування платформою дуже просте, якщо порівнювати з аналогічними інструментами, наприклад Jira. Ресурсом користуються більшість бізнесів.

Порівнюючи з іншими інструментами для менеджменту проектів, то Асана має базові та мінімалістичні функції, щоб не перезавантажувати систему.

Структура гнучка, тому додатком можуть користуватися фахівці з різних галузей.

Є безкоштовна версія, але з обмеженим доступом, або є платні версії - Premium та Enterprise.

Якщо використовувати безкоштовну версію додатка, то можливо додати лише 15 співробітників, а також доступ до деяких інструментів буде обмежений.

Преміум версія коштує 9,99 дол. США на місяць, в нього входить контроль адміністратора. Є доступ до всіх інструментів, та можна додавати безліч людей до команди.

Enterprise версія має ще більше можливостей, таких як:

- контроль адміністратора;
- підтримка кожного дня;
- індивідуальний брендинг;
- безпека та захист даних;
- міжрегіональне резервне копіювання.

Ці рівні приблизно збігаються з цінами конкуруючим Trello, який також пропонує схему ціноутворення freemium, починаючи з безкоштовного базового варіанту, бізнес-пакету середнього рівня вартістю 9,99 дол. І повного корпоративного плану. [13]

Переваги:

- Можливість розділити складний проект на сфери діяльності.
- Інтуїтивно зрозумілий інтерфейс.
- Інтеграція з Dropbox, Google Drive, Evernote і ін.
- Різноманітна функціональність і хороша деталізація.
- Завдання можна ставити в кілька рівнів.
- Система підходить для великих і малих проектів.
- Можливість ранжувати завдання по пріоритетності.
- Зручна система коментування та зв'язки з колегами.

- Є безкоштовна версія і програма Asana є в безкоштовному доступі для скачування на різні гаджети.

Недоліки:

- Немає діаграми Ганта, немає можливості відстежувати час, який витрачають учасники проекту.
- Сервіс не можна купити, так як він надається тільки в місячну оренду від 8 дол. США.
- Експорт документів. За відгуками, бачимо проблему - якщо треба скачати певні файли з сервісу, то немає стандартних форматів таких як PDF чи Excel.
- Призначити відповідального на завдання можна лише одну людину, через те, що компанія вважає, що це простіше і не будуть виникати проблеми щодо того, хто відповідає за проект.

ВИСНОВКИ ПО РОЗДІЛУ 1

Отже, проаналізувавши всі чотири інструменти для управління проектами, можна зробити висновок, що всі більш-менш схожі між собою, але відрізняються додатковими функціями та вартістю. Найбільш популярним сервісом серед бізнесів є Jira, через те, що в неї великий функціонал, зручний та сучасний інтерфейс та він більш призначений для бізнес-процесів. Redmine – схожий інструмент, до того ще і безкоштовний, але з меншим функціоналом та дуже простий. Trello та Asana виглядають однаково, у вигляді Kanban – дошки. На мою думку, вони більш підходять для повсякденного життя споживачів, для слідкування щоденних справ та планів на майбутнє, використовується як щоденник.

Якщо дивитися на попит, то бачимо що Jira займає лідируючі позиції на ринку, найменш популярним є Redmine (див.рис.1.6).

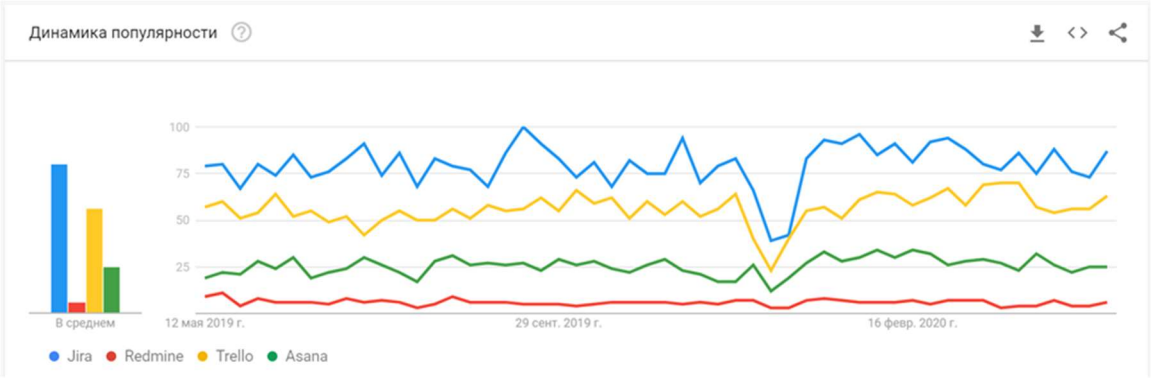


Рис.1.6. Попит на ресурси з управління проектами. Джерело: Google Trends

РОЗДІЛ 2
ОПИС ПРИНЦИПІВ, ТЕХНОЛОГІЙ ТА МОВ ПРОГРАМУВАННЯ

2.1 Архітектурне рішення

2.1.1 MVC

Архітектурні патерни існують для того, шоб уніфікувати архітектурні рішення різних розробників в одній команді. Коли до проекту потрапляє новий співробітник, то набагато легше пояснити що відбувається в окремих ділянках програми якщо існує єдиний піднід до написання програмного забезпечення.

Для свого проекту я вибрав архітектурний патерн MVC – model view controller, у перекладі «модель представлення контролер». З його назви одразу зрозуміло, що він включає в себе три модулі, на які ділить програмне забезпечення (рис 2.1). Його призначення розділити програму на рівень роботи з даними, рівень бізнес-логіки та рівень представлення користувачу.

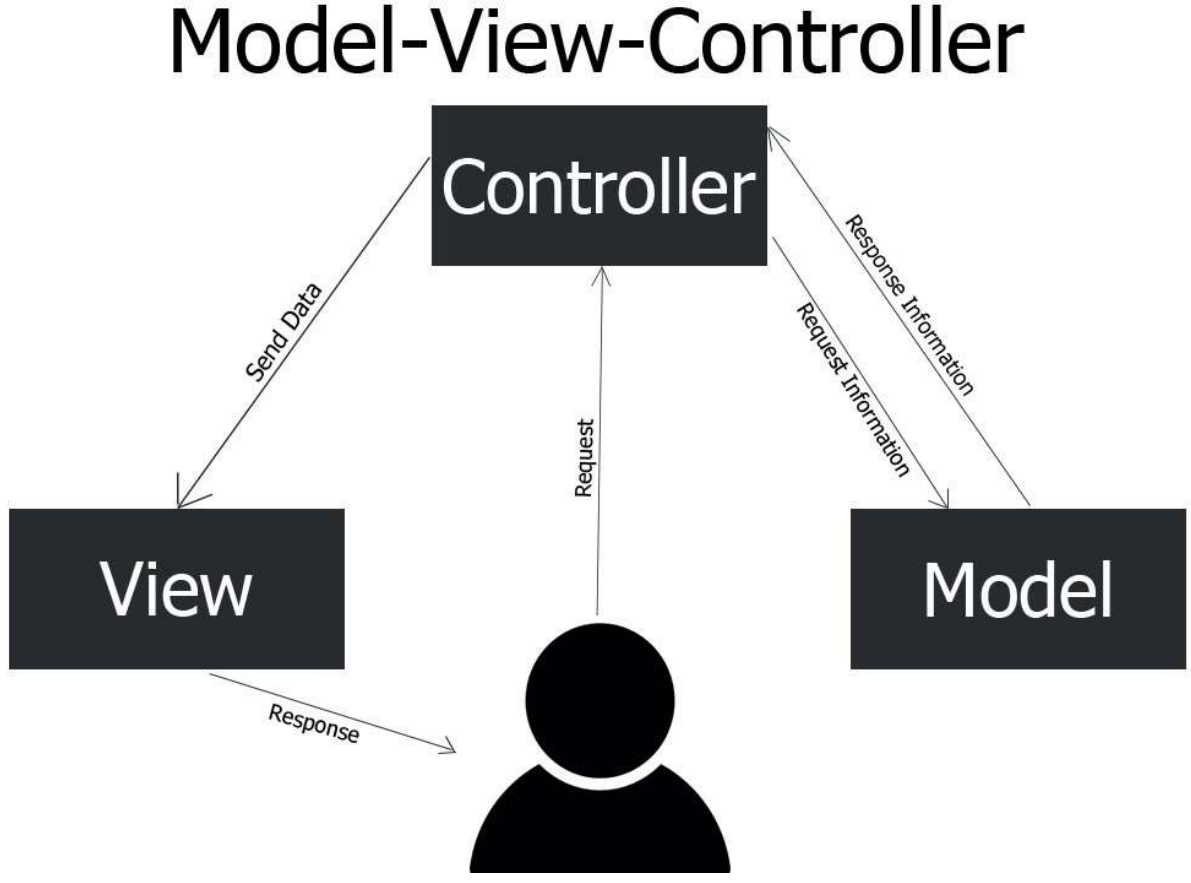


Рис 2.1. Схема MVC

Модель

Модель – забезпечує представлення даних. У даній частині програми повинні бути методи доступу до даних, їх зміна та створення. Модель не повинна залежати від представлення. Зазвичай робота с моделью зводиться до базових CRUD-методів:

- Create (Створити)
- Read (Прочитати)
- Update (Оновити)
- Delete (Видалити)

Представлення

Представлення – забезпечує представлення результату виконання програми. У сучасному світі ми користуємося сервісами з різних пристроїв, які потребують різного представення, хоча це може будти один сервіс, ті ж самі дані і функціонал. І за допомогою шаблону MVC ми можемо розробити модель і контроллер в єдиному екземплярі, а представленнь для користувачів безліч.

Контролер

Контролер – забезпечує виконання бізнес-логіки. Цей компонент виступає своєрідною зв'язуючою ланкою між моделью і представленням. Також у контролері відбувається програмування всіх бізнес-процесів. З рисунку 2.1 видно, що користувач подає запит до контролера, в свою чергу контролер бере потрібні йому дані у моделі, обробляє їх та передає на представлення, де ці дані перетворюються у зрозумілий для користувача вид, та надсилається йому відповідь.

Якщо взяти до уваги ту частину системи, яка розглядається у даній роботі, а саме прикладний програмний інтерфейс, і розглядати його як кінцевий продукт, то представлення може буде у вигляді одного з текстових форматів обміну даних, таких як JSON, XML і т.д.

2.1.2 REST API

REST API (Representational State Transfer Application Programming Interface) – архітектурний стиль взаємодії компонентів розподіленої системи у мережі. [15] Простими словами це набір правил, за якими будується прикладний програмний інтерфейс. Доступ до API отримується за допомогою HTTP-методів, які можна зіставити з CRUD-операціями:

- Http метод POST – create
- Http метод GET – read
- Http метод PUT – update
- Http метод DELETE – delete

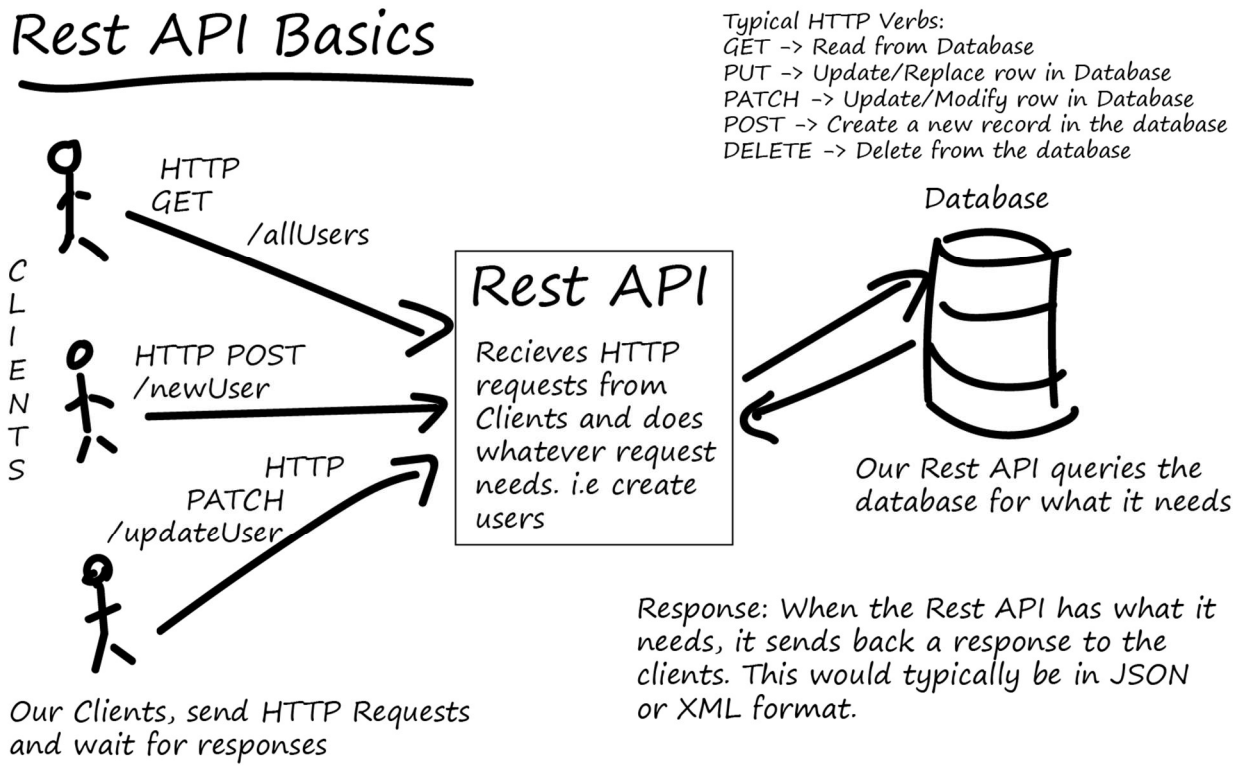


Рис 2.2. Схема REST API

Правила за допомогою яких будується REST API:

1. Клієнт-серверна модель.

Спочатку треба привести архітектуру до клієнт-серверної моделі, шоб додаток можно було розділити на дві частини, які можно розробляти незалежно. Але при виборі побудови застосунку по патерну MVC з цим проблем не буде.

2. Відсутність стану.

В період взаємодії клієнта із сервером, на останньому не зберігаєється ніяка інформація про клієнта. Стан сесісії зберігається лише на боці клієнта. Під час обробки запитів рахується, що клієнт знаходиться у перехідному стані. Кожен запит сервер повинен обробляти незалежно від попередніх запитів.

3. Кешування

Ще одним правилом REST є те, що система повинна підтримувати кешування, тобто дані, які передаються сервером, повинні містити інформацію про те, чи можна їх кешувати, і якщо можна, то як довго. Це дозволяє збільшити продуктивність, не роблячи повторних запитів, але треба фільтрувати інформацію, яку можна кешувати, щоб у користувача не залишалася інформація, яка може застаріти.

4. Однорідність інтерфейсу.

Наявність уніфікованого інтерфейсу є фундаментальним вимогою дизайну REST-сервісів. Уніфіковані інтерфейси дозволяють кожному з сервісів розвиватися незалежно.[15]

І якщо витримати ці правила, то система отримає такі переваги:

- Надійність (тому що серверна частина не зберігає дані про користувача, які можуть бути втрачені);
- Продуктивність (за рахунок використання кеша);
- Масштабованість;
- Простота інтерфейсів;
- Портативність компонентів;

- Легкість внесення змін;
- Здатність прилаштовуватись до нових вимог системи

2.2 Мова програмування

З оглядом на описані вище архітектурні рішення підходить об’єктно орієнтована мова програмування Java. Java - це мова програмування та обчислювальна платформа, вперше випущена компанією Sun Microsystems в 1995 році. Є безліч додатків і веб-сайтів для яких потрібно встановити JVM – віртуальну Java машину, будь то віддалений сервер або ваша локальна машина, шоб вони працювали. І щодня створюється все більше Java програм, тому що вони є швидкими, безпечними та надійними. Ноутбуки, центри обробки даних, ігрові консолі, наукові суперкомп'ютери, мобільні телефони і багато іншого - це все різні девайси на якіх працюють програми написані на Java. Тому що для їх відтворення, на машині користувача повинна бути встановлена лише JVM потрібної версії і програма буде працювати на будь-якій платформі. До цього всього Java можно завантажити безкоштовно. Середовище виконання Java (JRE) - це те, що ви отримуєте при завантаженні програмного забезпечення Java. JRE складається з віртуальної машини Java (JVM), основних класів платформи Java та підтримуючих бібліотек платформи Java.

Переваги Java:

- Проста. Java полегшила життя, видаливши всі складності, такі як покажчики, перевантаження оператора, які можна побачити в C ++ або будь-якій іншій мові програмування.
- Портативна. Java не залежить від платформи, що означає, що будь-яка програма, написана на одній платформі, може бути легко перенесена на іншу платформу.
- Об’єктно-орієнтована. Що завгодно може вважатися об'єктом, який має певний стан, поведінку і всі операції будуть виконуватись з використанням цих об'єктів.

- Безпечна. Весь код спочатку компілюється в байт-код, який не читається людиною. Java не використовує явного вказівника і запускає програми всередині так званої пісочної скриньки, щоб запобігти будь-якій діяльності з ненадійних джерел. Це дає змогу розробляти системи та програми, що не містять вірусів.

- Динамічна. Java має можливість адаптуватися до розвиваючого середовища, яке підтримує динамічний розподіл пам’яті, за рахунок чого витрата пам’яті зменшується та підвищується продуктивність програми.

- Поширена. Java надає функціонал, який допомагає створювати розподілені програми. Наприклад роблячи віддалений виклик методу (RMI), програма може викликати метод іншої програми або системи в мережі та отримати вихідний потік. Ви можете отримати доступ до файлів, звернувшись до методів з будь-якої машини в Інтернеті.

- Надійна. У Java є потужна система управління пам'яттю. Це допомагає усунути помилки, оскільки вона перевіряє код під час компіляції та виконання.

- Високопродуктивна. Java досягає високої продуктивності завдяки використанню байт-коду, який можна легко перевести на рідний машинний код. За допомогою компіляторів JIT (Just-In-Time) Java забезпечує високу продуктивність.

- Інтерпритована. Java компілює початковий код в байт-код, який у свою чергу інтерпретується середовищем виконання Java, а саме віртуальною Java машиною (рис 2.3).[17]

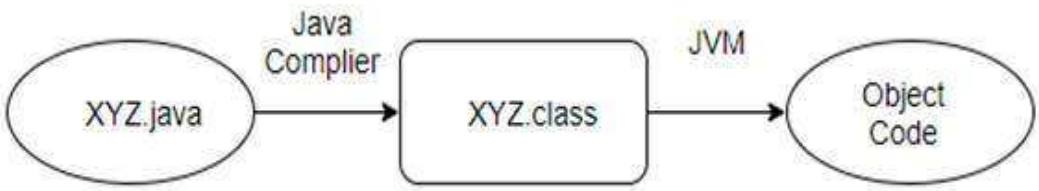


Рис 2.3. Схема виконання Java коду

- Багатопоточна. Java підтримує багатопоточне виконанне, включаючи набір примітивів для синхронізації. Це значно спрощує програмування з потоками.

Компоненти Java:

- JVM (Віртуальна Java машина). Це абстрактна машина. Це специфікація, яка забезпечує середовище виконання часу, в якому може виконуватися байт-код Java. З цього випливає три позначення:
    - Специфікація. Це документ, який описує реалізацію віртуальної машини Java. Її надають Sun та інші компанії.
    - Імплементація. Це програма, яка відповідає вимогам специфікації JVM.
    - Екземпляр виконання. Екземпляр JVM створюється кожного разу, коли ви пишете java команду в командному рядку та запускаєте клас.
- JRE (Сердовище викононня Java). JRE посилається на середовище виконання, в якому байт-код Java може бути виконаний. Він реалізує JVM (Віртуальна Java машина) і надає всі бібліотеки класів та інші файли підтримки, які JVM використовує під час виконання. Отже, JRE - це програмний пакет, який містить те, що потрібно для запуску програми Java. В основному, це реалізація JVM, яка існує фізично.
- JDK (Набір розробника Java). Це необхідний інструмент для:
    - Компіляції
    - Документації
    - Пакування Java програм
- JDK повністю включає JRE, яке містить інструменти для розробників на мові програмування Java (рис. 2.4). Набір розробника Java надається безкоштовно. Поряд з сердовищем викононня Java, він включає інтерпретатор, завантажувач класів, компілятор (javac), архіватор (jar), генератор документації

(javadoc) та інші інструменти, необхідні в розробці програм на Java. Словом, він містить сердовище викононня Java та інструменти для розробників.[16]

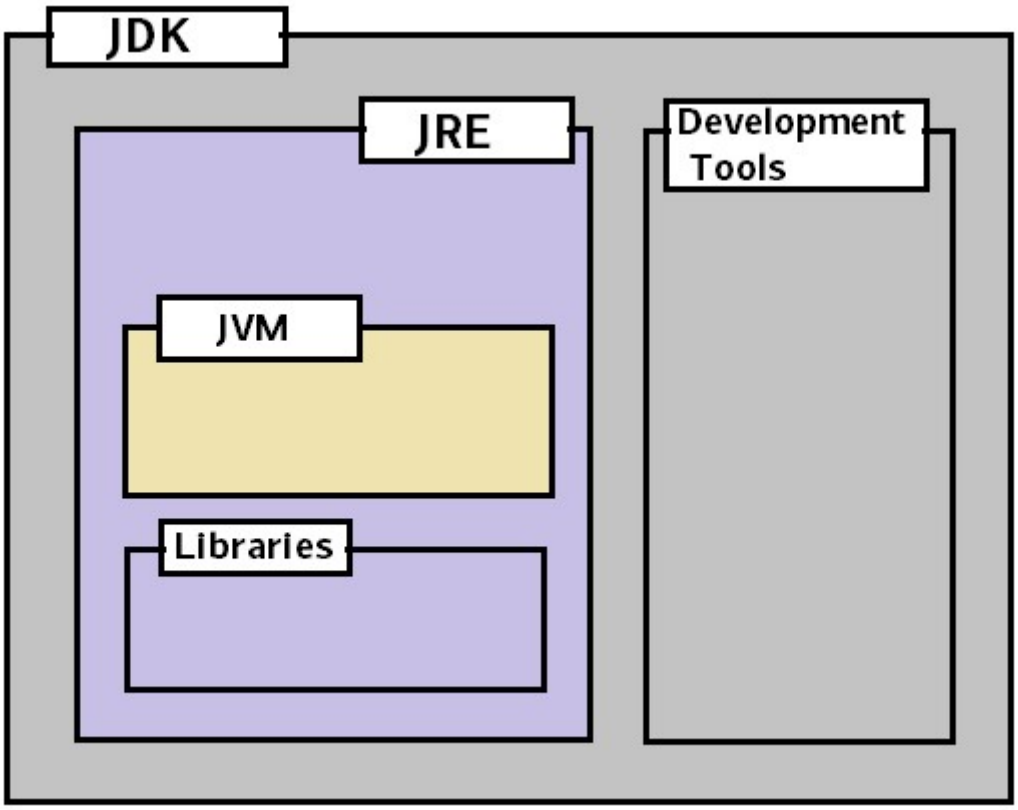


Рис 2.4. Схема відмінності JDK, JRE, JVM

Також треба розповісти як працює ця «магічна скриня» під назвою JVM. Вона працює як двигун для запуску програм, написаних на Java. Віртуальна машина викликає метод входу у програму – «main».

Коли ми збираємо .java документ, компілятор Java створює файли .class з подібними іменами, присутніми в назві .java документу. Цей файл з розширенням .class переходить у інші стани, коли ми його запускаємо (рис. 2.5). Ці всі засоби, разом, описують всю JVM.

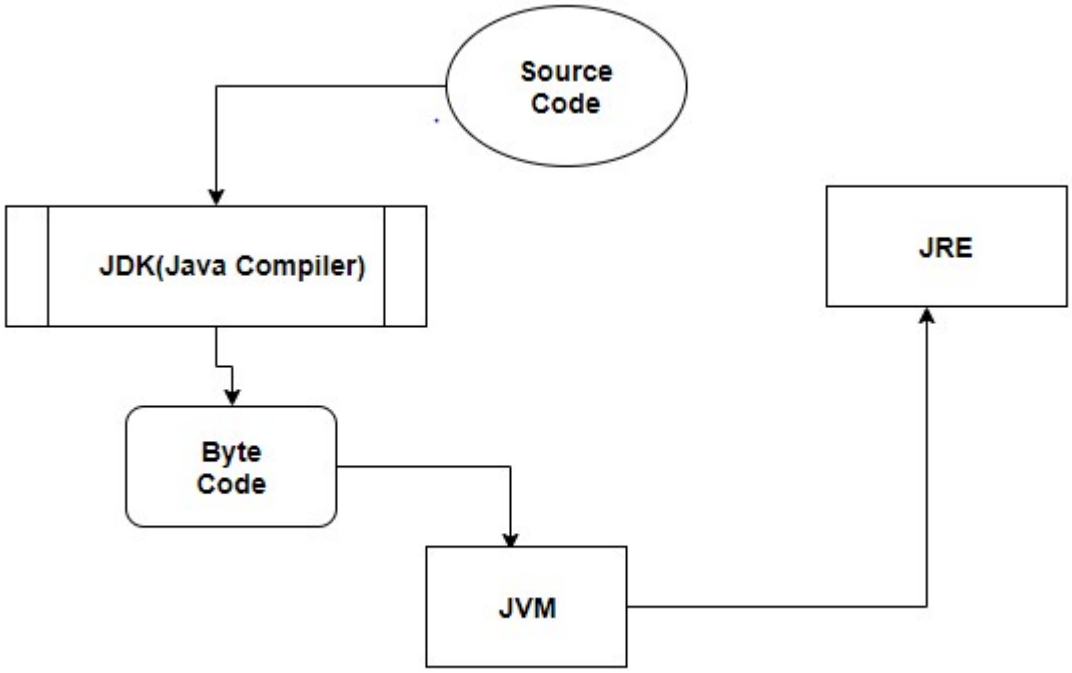


Рис 2.5. Схема відмінності JDK, JRE, JVM

JVM надає ідеальне виконання для Java-програм, використовуючи багато пропагованих стратегій, поєднуючи гарну модель пам'яті, збірник сміття та універсальний, адаптивний оптимізатор.

Віртуальна машина Java саме «віртуальна», тому що надає машинний інтерфейс, який не спирається на основні робочі рамки та конструкцію машинного обладнання. Ця свобода від обладнання та робочої структури є нахідкою для твердження «написана один раз, запущена деінде».[17]

Завантажувач класів - це підсистема, яка використовується для складання файлів .class. Він виконує три примітні потужності, тобто завантаження класів, зв'язування та ініціалізація (рис.2.6).

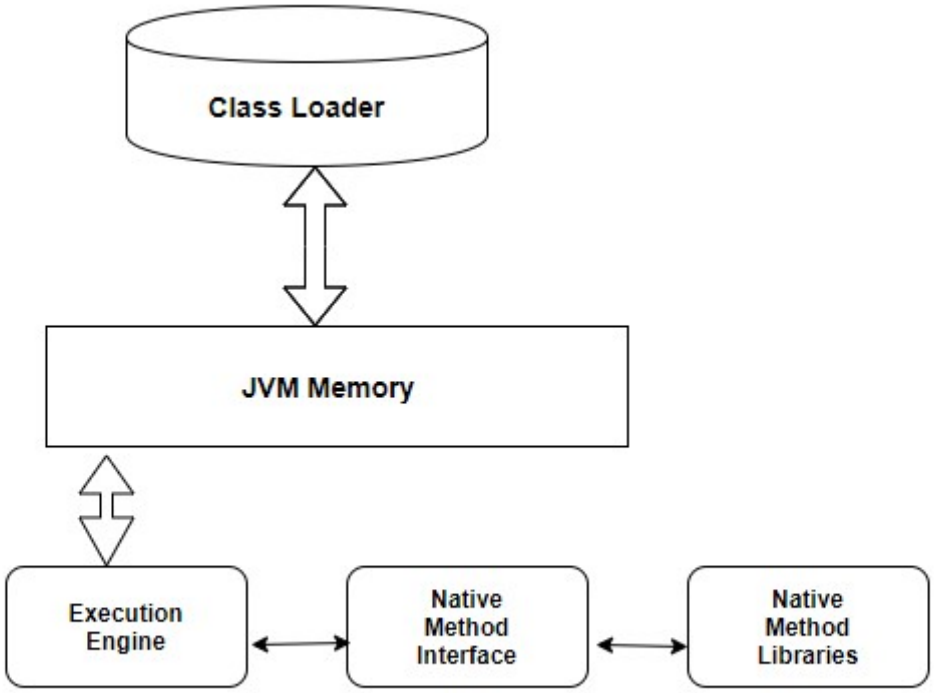


Рис 2.6. Архітектура JVM

2.3 Бібліотеки та фреймворки

2.3.1 Java Collection Framework

Java Collection Framework - ієрархія інтерфейсів і їх реалізацій (рис. 2.7), яка є частиною JDK і дозволяє розробнику користуватися великою кількістю структур даних з «коробки».[18]

Java Collection Framework був розроблений, щоб досягти декілька цілей:

- Фреймворк повиннен бути високопродуктивними. Реалізації для основних колекцій (динамічні масиви, зв'язні списки, дерева та хеш-таблиці) мають бути високоефективними.
- Фреймворк повинен дозволяти колекціям різних типів працювати в аналогічній манері та з високим ступенем сумісності.
- Фреймворк повиннен легко розширити та адаптувати колекцію.

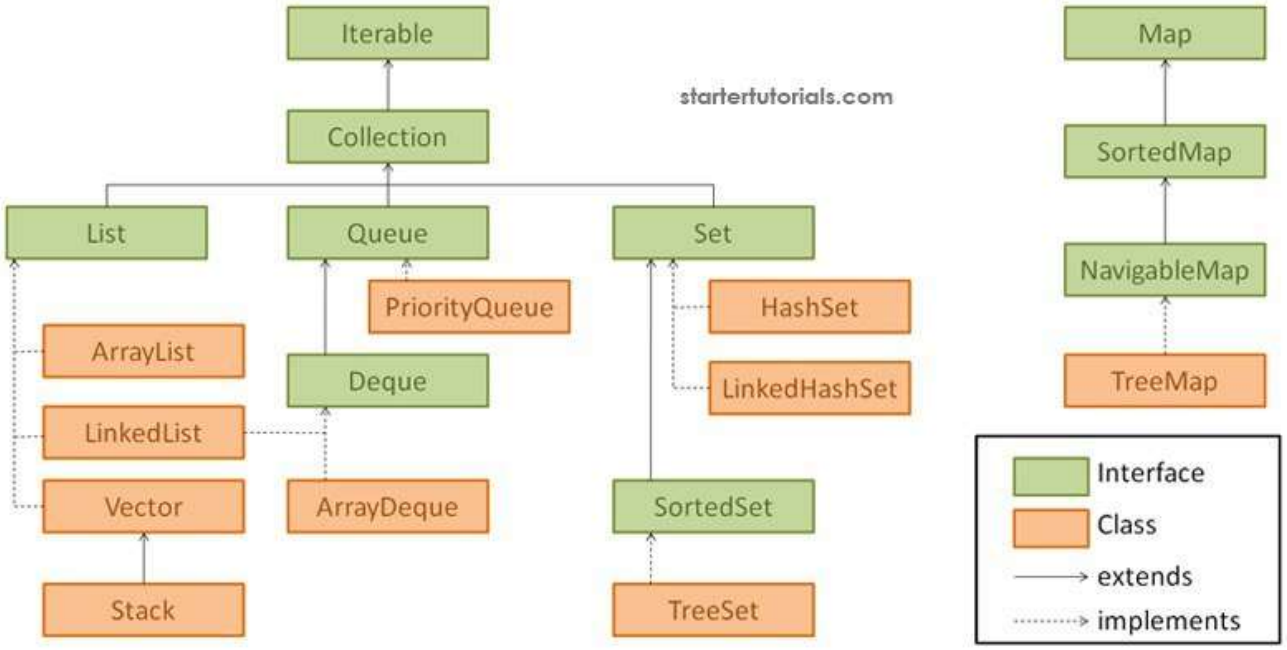


Рис 2.7. Java Collection Framework - ієрархія інтерфейсів і їх реалізацій [20]

Для досягнення цих цілей, весь фреймворк колекцій розроблений навколо множини стандартних інтерфейсів. Кількома стандартними реалізаціями, такими як LinkedList, HashSet і TreeSet, цих інтерфейсів, передбачено, що ви можете використовувати як власне їх, так і власноруч реалізувати колекцію яку вам потрібно.

Java Collection Framework - це єдина архітектура для представлення та маніпулювання колекціями. Усі фреймворки колекцій містять наступне:

- Інтерфейси. Це абстрактні типи даних, які представляють колекції. Інтерфейси дозволяють маніпулювати колекціями незалежно від деталей їх представлення. В об'єктно-орієнтованих мовах інтерфейси, як правило, утворюють ієрархію
- Реалізації, тобто класи. Це конкретні реалізації інтерфейсів колекцій. По суті, вони є багаторазовими структурами даних.
- Алгоритми. Це методи, які виконують корисні обчислення, такі як пошук та сортування, на об’єктах, що реалізують інтерфейси колекції. Кажуть,

що алгоритми є поліморфними: тобто один і той же метод може бути використаний у багатьох різних реалізаціях відповідного інтерфейсу колекції.

В додаток до колекцій, фреймворк визначає кілька інтерфейсів словників та класів. Словники зберігають пари ключ-значення. Хоча словники не є колекціями при дослівному використанні терміна, але вони повністю інтегровані з колекціями.

Основним інтерфейсом, який є основою для всіх колекцій Java, є інтерфейс Collection. Цей інтерфейс описує основні методи роботи з кожною колекцією. Найбільш важливі методи показані на рис 2.8.

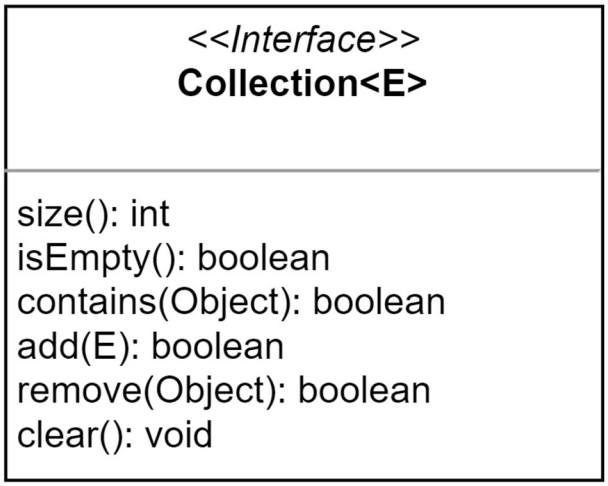


Рис 2.8. Основні методи інтерфейсу Collections

Пояснення основних методів інтерфейсу Collections:

- size() - повертає поточну кількість елементів колекції
- isEmpty() - повертає true, якщо в колекції немає елементів, інакше false
- contains(Object) - повертає true, якщо колекція містить елемент, переданий як параметр
- add(Object) - додає в колекцію новий елемент; повертає true, якщо колекція була змінена (елемент був доданий), інакше false

- addAll(Collections) - додає в список всі елементи іншої колекції
- toArray() - копіює елементи зі списку в масив
- remove(Object) - вилучає елемент із колекції; повертає true, якщо колекція була змінена (елемент існував і був видалений), інакше false
- removeAll(Collections) - видаляє зі списку всі елементи іншої колекції
- clear() - очищає вміст колекції

Інтерфейс колекції розширює інтерфейс Iterable. Цей інтерфейс визначає методи ітерації не тільки колекцій, але й усіх об'єктів, які можна перебрати. В інтерфейсі є метод iterator(), який всі колекції повинні реалізувати. Він повертає ітератор. Інтерфейс містить два методи з реалізацією за замовчуванням: forEach() та spliterator().

Ітератори - це об'єкти, які використовуються для ітерації через колекції.

Давайте детальніше розглянемо інтерфейс Iterator. Його реалізація повертається однойменним методом. Цей інтерфейс містить два важливі методи: next() та hasNext():

- next() - повертає наступний елемент
- hasNext() - повертає true, якщо наступний елемент існує

За допомогою цих двох методів є можливість переглядати колекцію від початку до кінця.

З версії Java 8, цей інтерфейс містить також такі методи:

- remove(Object) - вилучає елемент із колекції, якщо він підтримується колекцією, інакше буде кинуто UnsupportedOperationException; це єдиний правильний спосіб видалити елемент із колекції під час ітерації по ньому
- forEachRemaining(Consumer) - проходить кожен елемент колекції і застосовує до нього певну дію

Інтерфейс Collection розширюється іншими методами інтерфейсів List, Set та Queue на основі використання. Інтерфейс Map, який містить методи роботи з колекціями типів ключ-значення, є абсолютно окремим. Основні методи цих інтерфейсів реалізовані в абстрактних класах, відповідно до типу інтерфейсу: AbstractList, AbstractSet, AbstractQueue та AbstractMap. Абстрактні класи використовуються тому, що деякі конкретні реалізації інтерфейсу можуть поділяти реалізацію основних методів (size(), isEmpty()), але також матимуть різні методи, такі як add(), remove(). Крім того, ці абстрактні класи корисні у тому випадку, коли ми хочемо реалізувати власну колекцію, але хочемо, щоб основа вже була реалізована.

Щоб бути абсолютно точним, всі перераховані вище абстрактні класи, крім AbstractMap, також успадковують від загального абстрактного класу AbstractCollection. Усі класи можна знайти в пакеті java.util. Ці класи мають одну загальну особливість: вони не є безпечними для потоків. Це означає, що змінювати їх елементи з декількох потоків не є безпечним. У Java цю проблему вирішують за допомогою класів, які знаходяться в пакеті java.util.concurrent. Існують також однойменні класи з підтримкою безпеки для багатопотокових потоків. Наприклад, для ArrayList існує безпечна для потоків версія під назвою CopyOnWriteArrayList.[21]

Перед тим як розглянути більш детально реалізації Collections фреймворка, треба розповісти про звивачні масиви у мові програмування Java, і чому їх потрібно покращити. Однією з основних характеристик масиву є те, що він має фіксовану кількість елементів. З цієї причини він навіть не розглядається як колекція у деяких джерелах, оскільки він не відповідає частині його визначення. Елементи в масиві індексуються цілими числами, починаючи з нуля.

Основним недоліком використання масивів є те, що ми не можемо додавати або видаляти елементи під час виконання. Але у моєму проекті це часто потрібно робити. Однак є ситуації, коли масив - ідеальний вибір. Цей

"недолік" - це ціна, яка сплачується за високу швидкість, що постачається за доступ до елементів масиву. Оскільки дані є одного типу (абсолютно однакові або успадковуються у загального класу), вони займають однаковий простір в пам'яті. Окремі елементи масиву зберігаються в пам'яті як безперебійна послідовність (рис 2.9).[22]

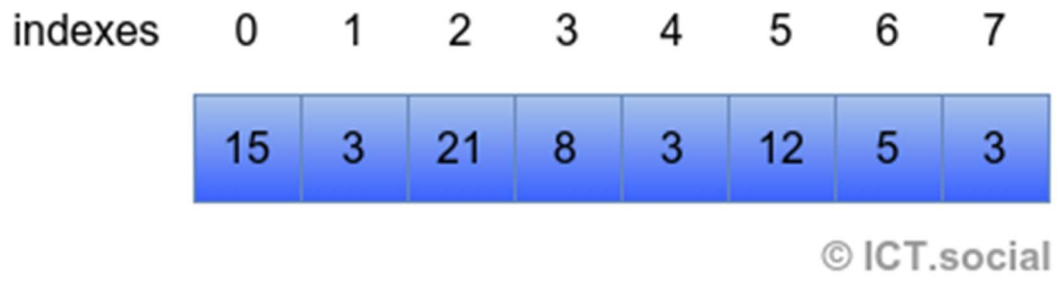


Рис 2.9. Представлення масиву

Списки - це колекції, які дозволяють нам додавати та видаляти елементи під час виконання. Вони можуть бути індексовані цілими числами як масиви, але цього не потрібно. В основному використовують два типи списків: ArrayLists та LinkedList.

Перевага списків у тому, що вони використовують той факт, що, хоча ми не можемо змінити розмір масиву під час виконання, ми можемо створювати нові масиви під час виконання.

У цьому випадку, список - це клас, який містить способи додавання та видалення елементів (та багато інших корисних методів, які нам наразі ми не розглядаємо). Клас, по суті, обертає масив і містить додаткову змінну, де зберігається кількість елементів. Коли створюється екземпляр, всередині створюється масив заданої кількості елементів, а змінна, що містить кількість елементів, встановлюється 0. Коли ми додаємо перший елемент, він зберігається в 0-му індексі в масиві та в кількість елементів збільшується. Ми зможемо додавати так елементи, поки не заповнимо масив. Після того, як масив

заповнений, ми просто створимо новий масив, скажімо, вдвічі більший (насправді розмір нового масиву розраховується за формулою, але для розуміння концепції це неважливо. Ми копіюємо елементи зі старого масиву в новий масив, а потім видаляємо старий. Як тільки цей новий масив буде заповнений, процес повториться. Колекція ArrayList, саме так внутрішньо і працює.

Список на рисунку 2.10 містить вісім елементів. Елементи зберігаються у внутрішньому масиві розміром 12 елементів. Останні чотири елементи не використовуються, а зовні виглядають так, ніби їх навіть немає.

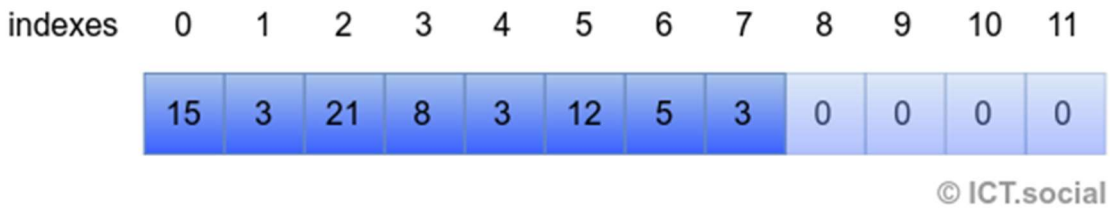


Рис 2.10. Представлення ArrayList

Перевага ArrayList у швидкому випадковому достуі до елементів за допомогою числових індексів завдяки внутрішньому масиву. Недоліком є час, необхідний для створення нового масиву та копіювання елементів у нього, хоча цей процес відбувається не надто часто. Ще один менш значимий недолік - це те, що колекція займає більше пам’яті, ніж потрібно. У будь-якому випадку, цей тип списку є найбільш використовуваною колекцією у мові програмування Java і досить добре оптимізований.

Крім того, List має такі методи:

- indexOf(Object) - повертає індекс першої появи даного елемента у списку
- lastIndexOf(Object) - аналогічно методу indexOf(Object), повертає індекс останньої появи даного елемента у списку

- removeIf(Predicate) - видаляє всі елементи, що відповідають заданій умові
- sort(Comparable) - сортує список. Важливо, щоб його елементи реалізували інтерфейс Comparable, інакше метод видає помилку. Основні класи та структури Java реалізують інтерфейс Comparable, для власних класів ми маємо реалізувати його самостійно[22]

Другий підхід для створення списків зі змінною кількістю елементів - це створення зв'язаного списку. Цей спосіб зберігати елементи не включає масиви і натомість, використовує інший принцип. Окремі елементи зв'язаного списку розміщуються випадковим чином у пам'яті (вони вже не є послідовністю) і кожен наступний елемент посилається один на одного. Ми можемо бачити це як ланцюжок, де перший елемент з'єднаний з другим, другий - з третім тощо (рис 2.11). Цей тип списку називається однозв'язаним.

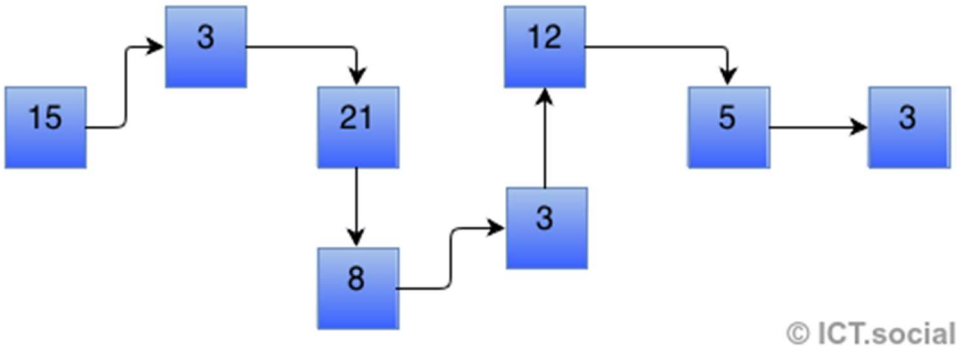


Рис 2.11. Представлення однозв’язаного списку

Якщо немає реальної причини для збереження пам'яті, то зазвичай посилаються на послідовні елементи в обох напрямках. Наприклад другий елемент посилається на первший і третій. У такому випадку ми маємо справу із двозв’язаним списком (рис 2.12).

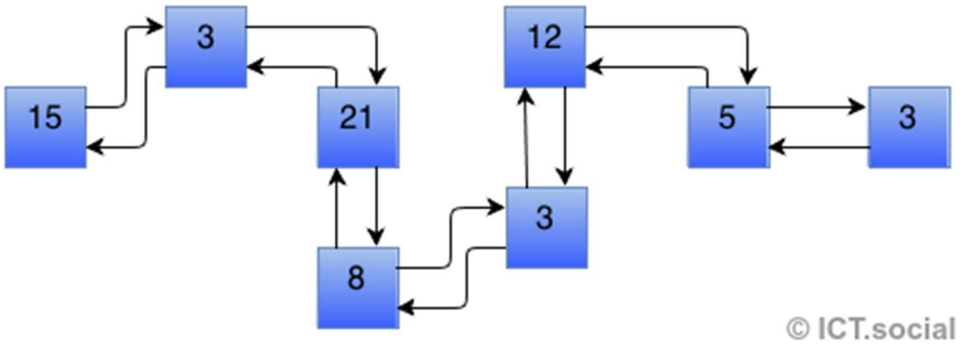


Рис 2.12. Представлення двозв’язаного списку

За допомогою зв’язаних списків ми втрачаємо можливість швидко отримувати доступ до будь-якого елемента за його індексом через те, що елементи не підряд лежать в пам'яті. Немає можливості ефективно перейти прямо до 100-го елемента та прочитати його значення. Якщо ми хочемо отримати доступ до певного елемента, нам доведеться перейти від першого елемента до другого, від другого до третього і продовжувати, поки ми не дійдемо до потрібного нам елемента. Складність часу читання та запису за заданим індексом залежить від кількості елементів у списку.

Однак іноді у нас немає потреби в індексації елементів, і тоді цей тип колекції стає дуже корисним. Повертаючись до вище написаного, можна пригадати, що LinkedList не використовує масиви. Тому більше не обмежені довжиною списку, і можемо додавати або видаляти елементи під час виконання, поки у нас є достатня кількість пам'яті. Також можемо видалити елементи в середині списку або додати нові елементи між існуючими. Використовуючи ArrayList, додавання нового елемента в середину було можливим лише в тому випадку, якщо змістити всі елементи праворуч, щоб звільнити місце для нового елемента. Це коштувало значної кількості часу, що залежить від кількості елементів. Що стосується зв’язаних списків, все, що потрібно зробити, - це змінити посилання на новий елемент між двома існуючими елементами (на інші елементи це не впливає).

Зв'язані списки представлені в Java загальною колекцією LinkedList. Цей клас використовує двозв'язані списки. Ми могли б їх запрограмувати, але це не варто того, оскільки працювати з ними складніше, а збереження пам’яті не дуже важливо для наших намірів і цілей.

LinekedList не містить в собі безпосередньо елементи, як це робив ArrayList. Натомість він зберігає елементи типу Node. Це вузли, які вказують один на одного і мають властивість елемента. Тут зберігається наш елемент, обгорнутий у вузол. Вузли розширюють наші елементи посиланнями на навколишні елементи. Давайте коротко розглянемо інші методи, які надає LinkedList порівняно з ArrayList:

- addFirst(Object) – додає новий елемент на початок списку
- addLast(Object) – додає новий елемент на кінець списку
- getFirst() – повертає перший елемент
- getLast() – повертає останній елемент
- removeFirst() – видаляє перший елемент
- removeLast() – видаляє останній елемент

В основному ми використовуємо зв'язані списки, коли хочемо додавати та видаляти елементи в середині списку, що буде дуже повільно, використовуючи ArrayList.[23]

Інтерфейси Set та List досить схожі між собою. Обидва інтерфейси являють собою сукупність елементів. Однак є деякі суттєві відмінності. Ці відмінності відображаються у методах, які містять інтерфейси Set та List.

Перша відмінність між інтерфейсом Set та List полягає в тому, що один і той же елемент не може зустрічатися більше одного разу в Set. Це відрізняється від List, де кожен елемент може повторюватися.

Друга відмінність інтерфейсів Set і List полягає в тому, що елементи у множині не мають гарантованого внутрішнього порядку. Елементи у списку мають внутрішній порядок, і вони можуть бути відтворені в тому ж порядку.

Розширюючи інтерфейс Collection, всі методи в інтерфейсі Collection також доступні в інтерфейсі Set.

У Java Collections API можна вибрати одну із наступних реалізацій множини:

- EnumSet
- HashSet
- LinkedHashSet
- TreeSet

Кожна з цих реалізацій множини поводиться дещо по-різному щодо порядку елементів під час ітерації, та часу (позначення велике О), необхідного для вставки та доступу до елементів у колекціях.

HashSet базується на HashMap. Це не дає гарантій щодо послідовності елементів, коли ви їх від. Зазвичай використовують HashSet, якщо потрібно отримувати доступ до елементів випадковим чином. Це тому, що для доступу до елементів хеш-таблиці використовуються хеш-коди.

Хеш-код елемента - це унікальний ідентифікатор, який допомагає ідентифікувати елемент у хеш-таблиці.

HashSet не може містити повторюваних елементів. Отже, кожен елемент набору має унікальний хеш-код.[24]

LinkedHashSet відрізняється від HashSet тим, що гарантує, що порядок елементів під час ітерації такий же, як і порядок, при якому вони були вставлені в LinkedHashSet. Повторне вставлення елемента, який вже є у LinkedHashSet, не змінює цей порядок. Елементи LinkedHashSet зберігаються в хеш-таблицях,

схожих на HashSet. Однак пов'язані хеш-набори підтримують внутрішньо двозв’язаний список для всіх його елементів. Двозв'язаний список визначає порядок введення елементів у хеш-таблиці. Через цю функціональність LinkedHashSet займає більше місця в пам’яті ніж HashSet, і має нижчу продуктивність.

TreeSet також гарантує порядок елементів під час ітерації, але порядок - це порядок сортування елементів. Іншими словами, порядок елементів, якщо використовувати Collections.sort() із списком або масивом, що містить ці елементи. Цей порядок визначається або їх природним порядком (якщо вони реалізують інтерфейс Comparable), або конкретною реалізацією Comparator. Якщо порівнювати цю реалізацію з LinkedHashSet, то остання має вище продуктивність, за рухнок того, що при кожній вставці елемента у TreeSet потрібно виконувати операцію сортування.

У пакет java.util.concurrent також додані реалізації, але функціонал цього пакету розбирати не будемо, так я він не використовувся у дипломному проекті.

Є два способи ітерації елементів Set:

- Використовуючи ітератор, отриманий із Set
- Використовуючи цикл

Як згадувалося раніше, при ітерації елементів у множині порядок елементів залежить від того, яку реалізацію Set використовувати.

Щоб пройтись по елементам набору за допомогою ітератора, спершу потрібно отримати ітератор із Set. Ітератор можна отримати, викликаючи метод iterator() з екземпляру своєї множини.

Інтерфейс Set реалізує інтерфейс Iterable. Ось чому є можливість перебрати елементи набору, використовуючи цикл for.[24]

Також інтерфейс Set дозволяє нам виконувати основні математичні операції над множинами, такі як об'єднання, перетин та підмножина:

- Об'єднання. Щоб отримати об'єднання двох множин x і y, можна використовувати x.addAll(y)
- Перетин. Щоб отримати перетин двох множин x і y, можна використовувати x.retainAll(y)
- Підмножина. Щоб перевірити, чи є x підмножиною y, ми можемо використовувати y.containsAll(x)

Інтерфейс Queue у Java являє собою структуру даних, розроблену для того, щоб елементи, при вставці додавалися у кінець черги, а при вилученні були отримані з початку черги (рис 2.13). Це схоже на те, як працює черга в супермаркеті.

Рис 2.13. Представлення черги

Інтерфейс Java Queue – це підтип інтерфейсу Collection. Він представляє впорядковану послідовність об'єктів, подібно до List, але його призначення трохи відрізняється. Оскільки інтерфейс черги Java є підтипом інтерфейсу Collection, усі методи інтерфейсу колекції також доступні в інтерфейсі черги.

У Java ми можемо знайти безліч реалізацій Queue. Їх можна класифікувати за двома наступними типами:

- Обмежені черги
- Не обмежені черги

Обмежені черги - це черги, обмежені місткістю, що означає, що нам потрібно надати максимальний розмір черги на момент створення. Наприклад ArueBlockingQueue.

Не обмежені черги - це черги, які не обмежені місткістю, це означає, що ми не повинні надавати розмір черги. Наприклад LinkedList.

Усі черги, доступні у пакеті java.util – не обмежені, а черги, які доступні у пакеті java.util.concurrent – обмежені.

Ще Queue можна класифікувати у наступні два типи:

- Блокуючі черги
- Неблокуючі черги

Усі черги, що реалізують інтерфейс BlockingQueue – блокуючі, а решта - неблокуючі.

BlockingQueues блокуються, поки вони не закінчать роботу або час не вичерпається, а не блокуючі черги так не роблять.

Деякі черги – двобічні, до них елементи можуть додаватись як на початок, так і в кінець.

У інтерфейса Queue найвживаніші реалізації це:

- LinkedList
- PriorityQueue

LinkedList – це досить стандартна реалізація черги. Елементи черги зберігаються всередині стандартної структури даних зв'язаного списку. Це дозволяє швидко вставити елементи в кінці (хвіст) списку та видалити елементи з початку (заголовка) списку. Більш детально ми розібрали цю реалізацію при описі інтерфейсу List.

PriorityQueue зберігає свої елементи внутрішньо відповідно до їх природного порядку (якщо вони реалізують інтерфейс Comparable) або згідно із Comparator, що передається до черги PriorityQueue.

Під час ітерації по черзі через її ітератор або через цикл for-each (який також використовує Iterator) послідовність, в якій ітераціюють елементи, залежить від реалізації черги.

Інтерфейс Queue у Java являє собою структуру даних, розроблену для того, щоб елементи, при вставці додавалися у кінець черги, а при вилученні були отримані з початку черги (рис 2.13). Це схоже на те, як працює черга в супермаркеті.

Так як інтерфейс Queue розширює інтерфейс Collection, то він має всі ті ж самі методи що List та Set. Хоча до них додаються ще свої:

- poll() – Отримує та видаляє заголовок цієї черги або повертає null, якщо ця чергу порожня.
- peek() – Отримує, але не видаляє, голову цієї черги, або повертає null, якщо ця черга порожня.
- offer(Object) – Вставляє вказаний елемент у цю чергу, якщо це можливо зробити негайно, не порушуючи обмежень пропускної здатності.
- element() – Отримує, але не видаляє, голову цієї черги.[25]

2.3.2 Spring Framework

Spring Framework спрощує створення корпоративних програм Java. Він надає все необхідне для використання мови Java у корпоративному середовищі, підтримуючи Groovy та Kotlin як альтернативні мови JVM та гнучкість для створення багатьох видів архітектури залежно від потреб програми. На момент виходу Spring Framework 5.1, Spring потребує JDK 8+ (Java SE 8+) та забезпечує підтримку «з коробки» JDK 11 LTS.

Spring підтримує широкий спектр сценаріїв застосування. На великому підприємстві додатки часто існують тривалий час і мають працювати на JDK та серверах додатків, цикл оновлення яких знаходиться поза контролем розробника. Інші можуть працювати як один jar-файл із вбудованим сервером, можливо, у хмарному середовищі. Ще іншими можуть бути автономні програми (наприклад, пакетні чи інтеграційні навантаження), яким не потрібен сервер.

Spring Framework є відкритим кодом. У ньому є велике та активне співтовариство, яке забезпечує постійний зворотний зв'язок на основі різноманітних випадків використання в реальному світі. Це допомогло Spring успішно розвиватися протягом дуже тривалого часу.

Термін "Spring" означає різні речі в різних контекстах. Він може бути використаний для позначення самого проекту Spring Framework, саме з цього і почалося все. З часом інші проекти Spring були побудовані на основі Spring Framework. Найчастіше, коли люди кажуть «Spring», вони мають на увазі всю сім’ю проектів.

Spring Framework поділен на модулі. Розробники можуть вибирати, які модулі їм потрібні. В основі лежать модулі основного контейнера, включаючи модель конфігурації та механізм введення залежності. Крім цього, Spring Framework надає фундаментальну підтримку різних архітектур прикладних програм, включаючи обмін повідомленнями, транзакційність даних та персистентність та веб. Він також включає серверну веб-структуру Spring MVC на основі сервлетів і, паралельно, реактивну веб-структуру Spring WebFlux.

Spring з'явився на світ у 2003 році як відповідь на складність ранніх специфікацій J2EE. Хоча деякі вважають, що Java EE та Spring є конкурентами, Spring фактично доповнює Java EE. Модель програмування Spring не містить специфікації платформи Java EE, швидше, вона інтегрується з ретельно підібраними індивідуальними специфікаціями Enterprise Edition:

- API сервлетів
- API WebSocket
- Багатопоточність
- JSON Binding API
- Bean валідація
- JPA
- JMS
- а також установки JTA/JCA для координації транзакцій, якщо це необхідно.

Spring Framework також підтримує специфікації Введення Залежностей та Загальні Аннотації, які розробники додатків можуть вибрати замість специфікацій Spring, передбачених Spring Framework.

На момент виходу Spring Framework 5.0, Spring вимагає як мінімум рівня Java EE 7 (наприклад, Servlet 3.1+, JPA 2.1+), і одночасно забезпечує інтеграцію «з коробки» з новішими API на рівні Java EE 8 (наприклад, Servlet 4.0, JSON Binding API). Це забезпечує Spring повністю сумісним, наприклад з Tomcat 8 і 9, WebSphere 9 і JBoss EAP 7.

З часом роль Java EE розвивалася в розробці додатків. У перші дні Java EE та Spring створювались додатки для розміщення на сервері. Сьогодні, за допомогою Spring Boot, програми створюються у дружелюбному, для devops спеціалістів та хмарних платформ, шлясі, із вбудованим Servlet контейнером. У версії Spring Framework 5, додаток WebFlux навіть не використовує безпосередньо API сервлетів і може працювати на серверах (таких як Netty), які не є контейнерами сервлетів.

Spring продовжує інновації та розвивається. Поза межами Spring Framework, існують інші проекти, такі як Spring Boot, Spring Security, Spring Data, Spring Cloud, Spring Batch. Важливо пам’ятати, що кожен проект має власне сховище вихідного коду, трекер завдань та правила випуску.

Коли ви дізнаєтесь про Spring Framework, важливо знати не тільки те, що він робить, але і які принципи він дотримується. Ось основні принципи Spring Framework:

- Забезпечити вибір на кожному рівні. Spring дозволяє відкладати дизайнерські рішення якомога пізніше. Наприклад, ви можете перемикати постачальників баз даних через конфігурацію, не змінюючи код. Те саме стосується багатьох інших інфраструктурних проблем та інтеграцій із сторонніми API.
- Визначити різноманітні точки зору. Spring охоплює гнучкість і вона не впевнена в тому, як слід робити. Це підтримує широкий спектр потреб у застосуванні з різними перспективами.
- Підтримувати міцну зворотну сумісність. Еволюцію Spring вдалося досягти декількома порушеннями змін між версіями. Spring підтримує ретельно підібраний діапазон версій JDK та сторонніх бібліотек для полегшення підтримки додатків і бібліотек, які залежать від Spring.
- Дбати про дизайн API. Команда Spring приділяє багато роздумів та часу для створення інтуїтивно зрозумілих API, які підтримують багато версій на багато років.
- Встановити високі стандарти якості коду. Spring Framework робить сильний акцент на змістовному, актуальному та точному javadoc (документація Java). Це один з дуже небагатьох проектів, який може вимагати чистої структури коду без кругової залежності між пакетами.

Якщо зараз починати робити проект за допомогою Spring, то можна почати використовувати Spring Framework, створивши додаток на базі Spring Boot. Spring Boot - це швидкий спосіб створити готовий Spring додаток. Він заснований на Spring Framework, підтримує правила щодо конфігурації та розроблений для того, щоб якомога шкидве розробляти.[26]

Особливості системи Spring, такі як IoC, AOP та управління транзакціями, роблять її унікальною серед переліку фреймворків. Деякі з найважливіших особливостей Spring Framework:

- IoC Container. Посилаючись на основний контейнер, який використовує шаблон DI (Dependency injection, введення заложностей) або IoC (Inversion of control, інверсія управління) для неявного надання посилання на об'єкт у класі під час виконання. Ця закономірність виступає альтернативою схемі локатора обслуговування. Контейнер IoC містить код, який обробляє управління конфігурацією об'єктів програми. У Spring Framework передбачено два пакети, а саме org.springframework.beans та org.springframework.context, що допомагає забезпечити функціональність контейнера IoC.
- Фреймворк доступу до даних. Дозволяє розробникам використовувати персистентні API, такі як JDBC та Hibernate, для зберігання даних у базі даних. Це допомагає у вирішенні різних проблем розробника, таких як взаємодія з підключенням до бази даних, як переконатися, що з'єднання закрите, як боротися з винятками та як реалізувати управління транзакціями. Це також дозволяє розробникам легко писати код для доступу до даних.
- Spring MVC framework. Дозволяє створювати веб-додатки на основі архітектури MVC. Усі запити, зроблені користувачем, спочатку проходять через контролер, а потім пересилаються на різні відображення, тобто на різні сторінки JSP або сервлети. Фунціонал обробки та валідації форм Spring MVC фреймворку можна легко інтегрувати з усіма популярними технологіями відображення, такими як ISP, Jasper Report, FreeMarker та Velocity.
- Управління транзаціями. Допомагає в управлінні транзакціями програми, не впливаючи на її код. Цей фреймворк пропонує API транзакцій Java (JTA) для глобальних транзакцій, керованих сервером, та локальних транзакцій, керованих за допомогою використання JDBC Hibernate, об'єктів даних Java (JDO) або інших API доступу до даних. Це дозволяє розробнику моделювати

широкий спектр транзакцій на основі декларативного та програмного управління транзакціями Spring.

- Spring Web Service. Генерує кінцеві точки та визначення веб-служб на основі класів Java, але керувати ними в додатку важко. Щоб вирішити цю проблему, Веб-сервіс Spring пропонує рівневі підходи, які окремо керується розширюваною мовою розмітки (XML). Spring забезпечує ефективне відображення XML-повідомлення на об’єкт і розробник з легкістю розподілить XML-повідомлення (об’єкт) між двома машинами.
- Рівень абстракції JDBC. Допомагає користувачам легко та ефективно керувати помилками. Код програмування JDBC може бути зменшений, коли цей рівень абстракції реалізований у веб-додатку. Цей шар обробляє винятки, такі як DriverNotFound. Усі SQLExceptions переведені в клас DataAccessException. Виключення доступу до даних Spring не є специфічним для JDBC, і тому об'єкти доступу до даних (DAO) не пов'язані лише з JDBC.
- Spring TestContext framework. Забезпечує засоби юніт та інтеграційного тестування для Spring додатків. Більше того, структура Spring TestContext забезпечує специфічні функції інтеграційного тестування, такі як управління контекстом та кешування DI тестових пристосувань та управління трансакційним тестом із семантикою відката за замовчуванням.

Spring Framework складається з семи модулів (рис. 2.14). Це модулі Spring Core, Spring AOP, Spring Web MVC, Spring DAO, Spring ORM, Spring контекст та Spring Web flow. Ці модулі забезпечують різні платформи для розробки різних корпоративних додатків; наприклад, можна використовувати модуль Spring Web MVC для розробки програм на базі MVC.

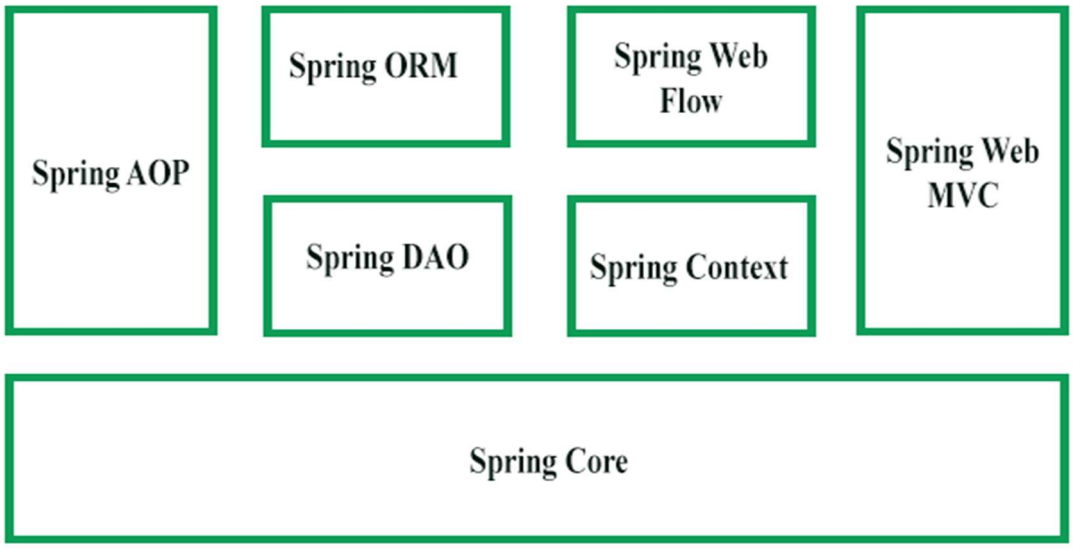


Рис. 2.14. Модулі Spring Framework

Детальніше розглянемо ці модулі:

- Spring Core. Модуль Spring Core, який є основною складовою структури Spring, забезпечує контейнер IoC. Існує два типи реалізації контейнера Spring, а саме: bean factory та application context. Bean factory визначається за допомогою інтерфейсу org.springframework.beans.factory.BeanFactory і діє як контейнер для бінів. Фабричний контейнер бінів дозволяє відокремити конфігурацію та специфікацію залежностей від логіки програми. У Spring Framework фабрика бінів виконує функцію центрального контейнера IoC, який відповідає за інстанціювання об'єктів програми. Він також налаштовує та збирає залежності між цими об'єктами. Існує чимало реалізацій інтерфейсу BeanFactory. Клас XmlBeanFactory - найпоширеніша реалізація інтерфейсу BeanFactory.
- Spring AOP. Подібно до об'єктно-орієнтованого програмування (ООП), яке розбиває програми на ієрархію об'єктів, AOP розбиває програми на аспекти. Модуль Spring AOP дозволяє реалізувати аспекти у Spring додатку, а аспекти -

це звичайні Spring біни або звичайні класи, що позначаються анотацією @Aspect. Ці аспекти допомагають в управлінні транзакціями, реєстрації даних та моніторингу помилок програми. Наприклад, управління транзакціями потрібно в банківських операціях, таких як переказ суми з одного рахунку на інший. Spring AOP модуль забезпечує рівень абстракції управління транзакціями, який можна застосувати до API транзакцій.

- Spring ORM. Модуль Spring ORM використовується для доступу до даних із баз даних у додатку. Він надає API для управління базами даних з JDO, Hibernate та iBatis. Spring ORM підтримує DAO, що забезпечує зручний спосіб побудови на базі DAO, ORM рішення.

- Spring Web MVC. Модуль Spring Web MVC реалізує архітектуру MVC для створення веб-додатків. Він розділяє код моделі та код представлення веб-програми. У Spring MVC, коли запит генерується з браузера, він спочатку переходить до класу DispatcherServlet (Front Controller), який відправляє запит до контролера (клас SimpleFormController або класу AbstractWizardformController), використовуючи набір обробників. Контролер витягує та обробляє інформацію, вкладену в запит, і передає результат класу DispatcherServlet у вигляді об'єкта моделі. Нарешті, клас DispatcherServlet використовує класи ViewResolver для передачі результатів у представлення, яке відображає ці результати для користувачів.

- Spring Web Flow. Модуль Spring Web Flow - це розширення модуля Spring Web MVC. Веб-структура Spring Web MVC забезпечує контролери, такі як клас SimpleFormController та клас AbstractWizardFormController, для реалізації заздалегідь заданого робочого процесу. Веб-потік Spring допомагає визначити XML-файл або Java клас, який керує робочим процесом між різними сторінками веб-програми. Весняний веб-потік поширюється окремо і його можна завантажити через веб-сайт http://www.springframework.org.

- Spring Web DAO. Пакет DAO у Spring Framework забезпечує підтримку DAO(Data Access Object, Об’єкт доступу до даних), використовуючи технології доступу до даних, такі як JDBC, Hibernate або JDO. Цей модуль вводить шар

абстракції JDBC, усуваючи необхідність у виснажливому кодуванні JDBC. Він також надає програмні, а також декларативні класи управління транзакціями. Spring DAO-пакет підтримує неоднорідне підключення до бази даних Java, що допомагає Spring працювати з декількома технологіями доступу до даних. Для легкого та швидкого доступу до ресурсів бази даних, Spring Framework пропонує абстрактні базові класи DAO. Для кожної технології доступу до даних, підтримуваної системою Spring, доступні декілька реалізацій. Наприклад, у JDBC клас JdbcDaoSupport та його методи використовуються для доступу до екземпляра DataSource та попередньо налаштованого екземпляра JdbcTemplate.

- Spring Application Context. Модуль Spring Application Context базується на модулі Core. Контекст програми org.springframework.context.ApplicationContext - це інтерфейс BeanFactory. Цей модуль отримує свою особливість з пакету org.springframework.beans, а також підтримує такі функції, як інтернаціоналізація, перевірка, розповсюдження подій та завантаження ресурсів. Контекст програми реалізує інтерфейс MessageSource та забезпечує функцію обміну повідомленнями для програми.[27]

ВИСНОВКИ ПО РОЗДІЛУ 2

На основі цих даних було прийнято рішення використати цей набір технологій для написання проекту.

Мову програмування Java було вибрано тому, що вона вже зарекомендувала себе як гарний інструмент для написання корпоративних рішень. Також великою перевагою є інтеграція із Spring Framework та всіма його модулями. За допомогою нього можна достатньо легко розробити прикладний програмний інтерфейс з дотриманням правил REST API. Також підтримка ORM надає інструменти роботи з базами даних. Це полегшує роботу з транзакціями, і не потрібно турбуватися за відкриття/закриття з'єднання за базою даних, обробку помилок, написання sql скриптів.

РОЗДІЛ 3

ІНСТРУКЦІЯ КОРИСТУВАЧА

Для початку роботи, на сервері потрібно запустити MySQL сервер і створити в ньому базу даних із кодуванням UTF-8. У файлі application.yml заповнити поля зображені на рисунку 3.1, згідно конфігурації бази даних створеної на сервері.

```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/diploma?serverTimezone=UTC
    username: user
    password: password
```

Рис. 3.1. Конфігурація бази даних у файлі application.yml

Далі потрібно запустити фреймворк для автоматизації збирання проектів maven так виконати команди mvn clean package. Після цього ми отримаємо файл з розширенням .war, він буде знаходитися у корневій директорії проекті, у папці target (рис. 3.2), який потрібно розгорнути на сервері. На сервері повинен працювати контейнер сервлетів tomcat. Наш .war файл повинен бути розміщений у робочій директорії, у папці webapps.

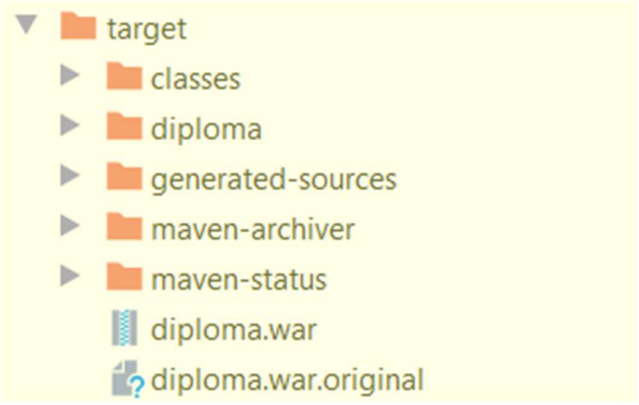


Рис. 3.2. Місцезнаходження файлу з розширенням .war

Після того як програма розгорнута на сервері і працює, можна приступати до роботи.

Так як у межах дипломної роботи було створено прикладний програмний інтерфейс, як частину системи розподілу завдань між розробниками, найкращим способом показати працездатність є swagger документація. Вона гарно візуалізує всі кінцеві точки, до яких можна звертатися із запитами, та прямо в ній можна ці запити виконати.

Спочатку ми бачимо поверхневий опис проекту (рис. 3.3).

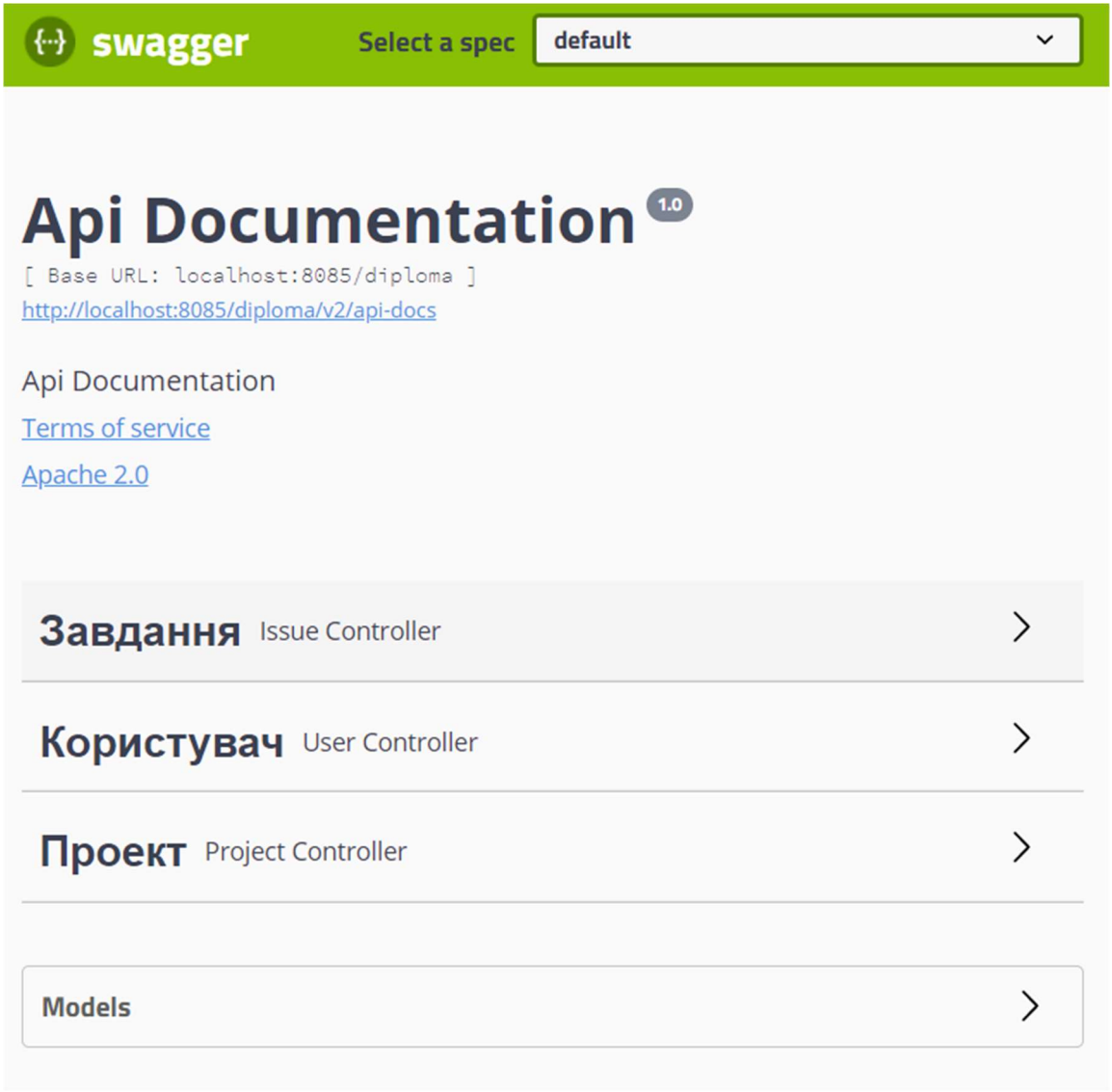


Рис. 3.3. Огляд проекту у swagger документації

Далі ми можемо розкрити кожен контроллер і подивитися його кінцеві точки та їх опис (рис 3.4).

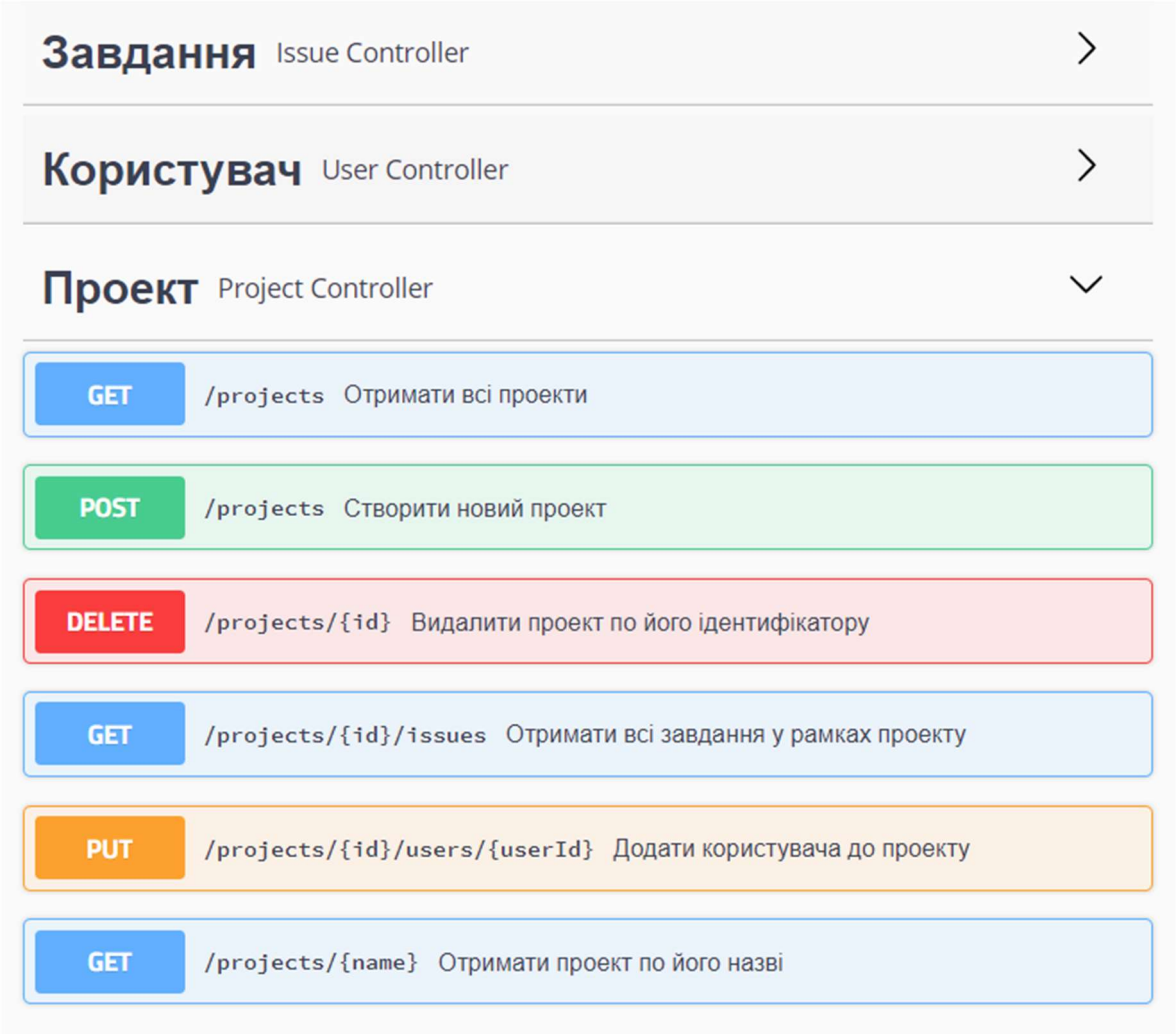


Рис. 3.3. Кінцеві точки Project Controller

Розкривши кожну кінцеву точку ми зможемо побачити які існують результати відповіді на наш запит, та при успішному виконанні можемо подивитися екземпляр структури відповіді (рис 3.4).

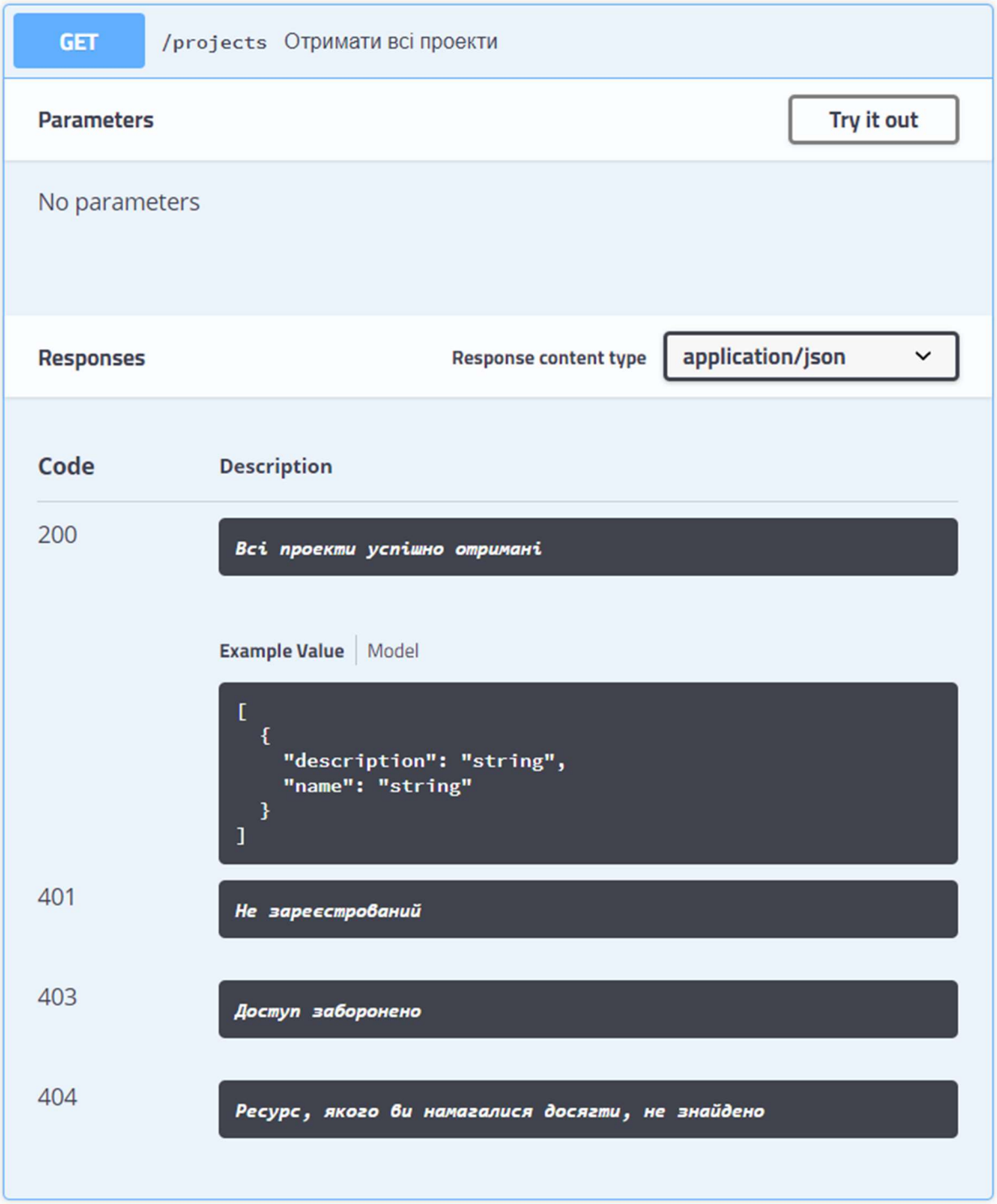


Рис. 3.4. Кінцева точка GET /projects

Тепер можемо відтворити можливий сценарій роботи програми. Як вже згадувалося вище, ми можемо це робити одразу у swagger документації.

Створимо проект. На рисунках 3.5 та 3.6 зображено запит та відповідь сервера відповідно.

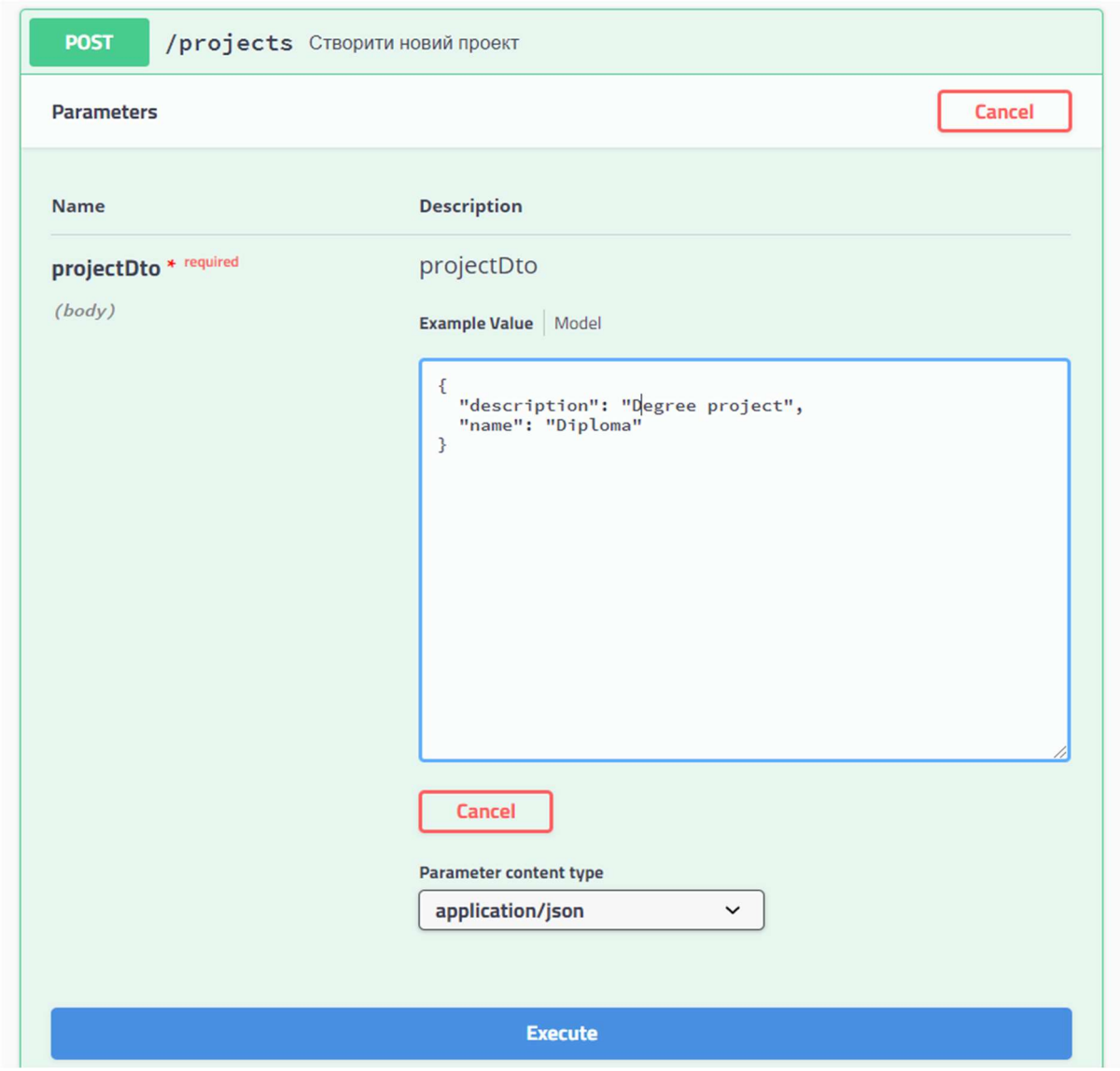


Рис. 3.5. Запит на створення нового проекту

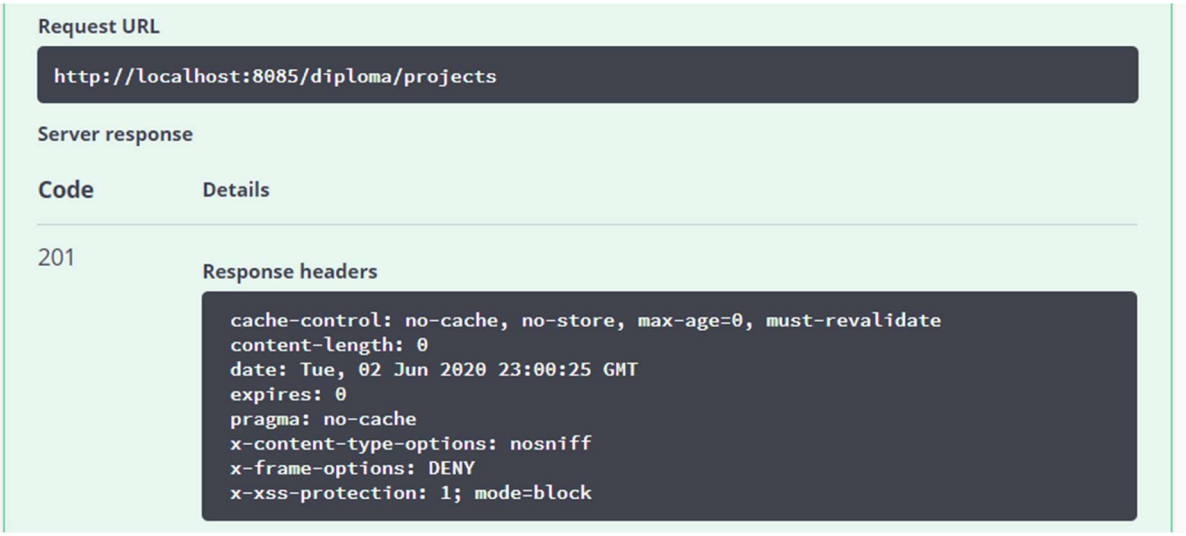


Рис. 3.6. Відповідь сервера на запит створення проекту

Далі треба створити нових користувачів (рис 3.7).

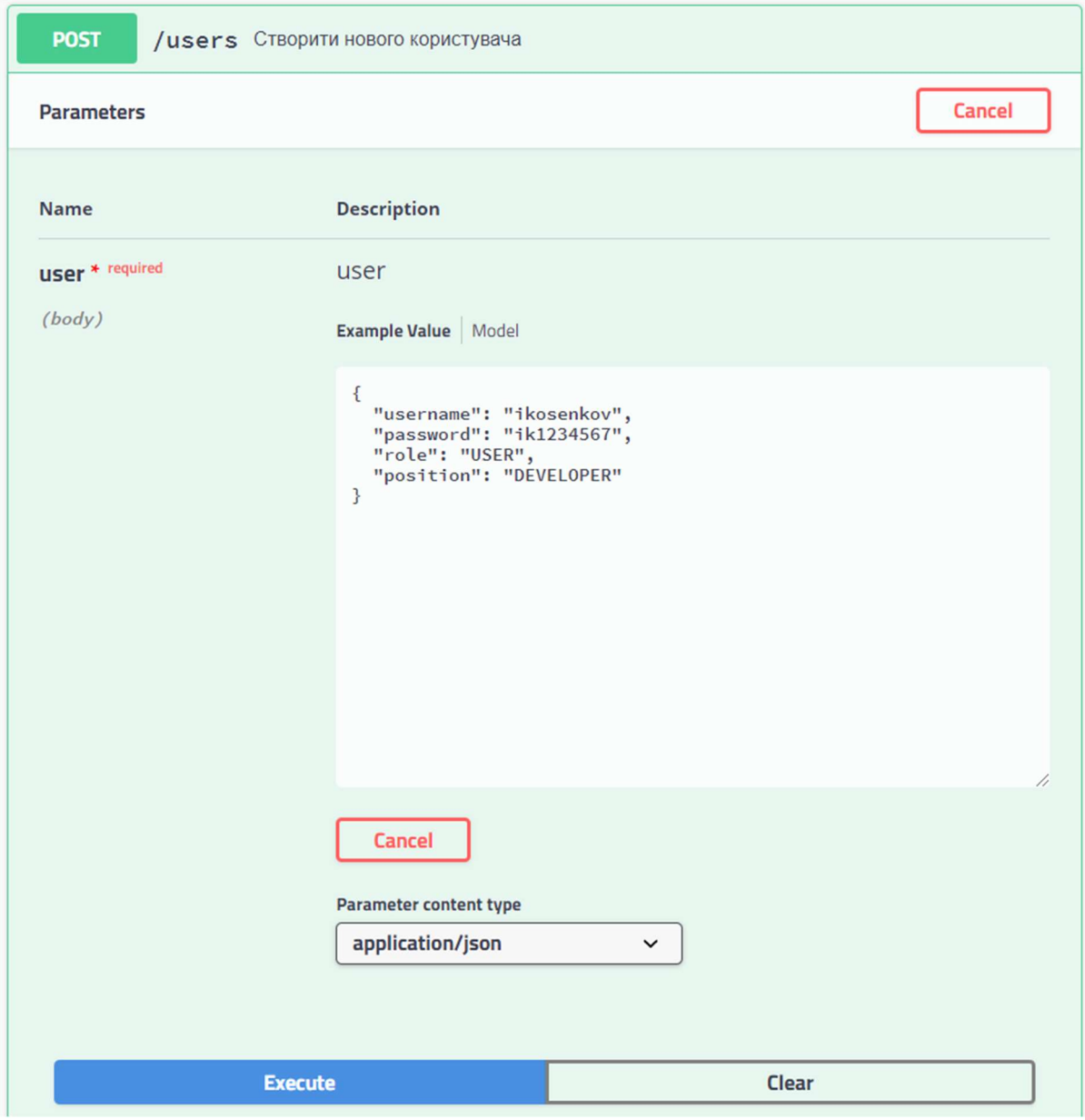


Рис. 3.7. Запит на створення нового користувача.

Після цього потрібно додати користувачів до проекту (рис. 3.8).

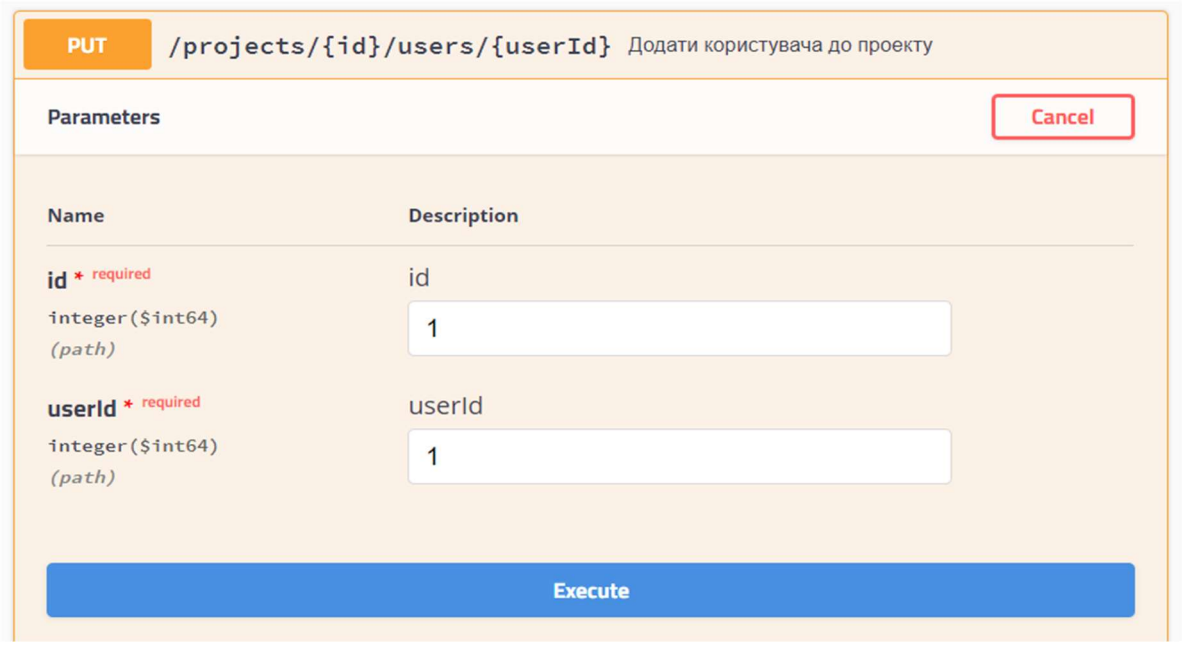


Рис. 3.8. Запит на додавання користувача до проекту.

Можемо подивитися загальний огляд проекту, і які користувачі зараз додані до нього (рис. 3.9).

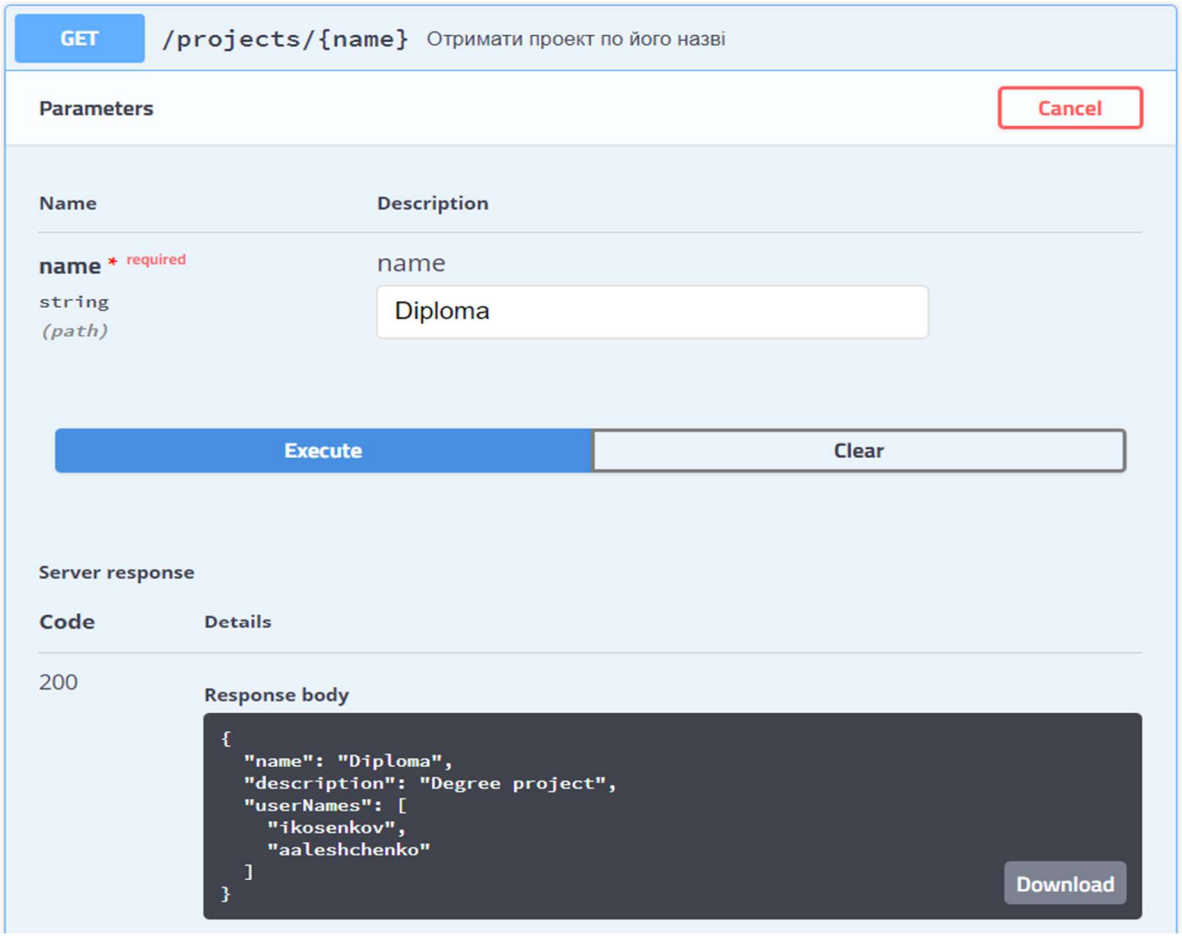


Рис. 3.8. Запит на отримання проекту по його назві.

Тепер створимо декілька завдань (рис 3.9) і назначимо на них користувачів(3.10).

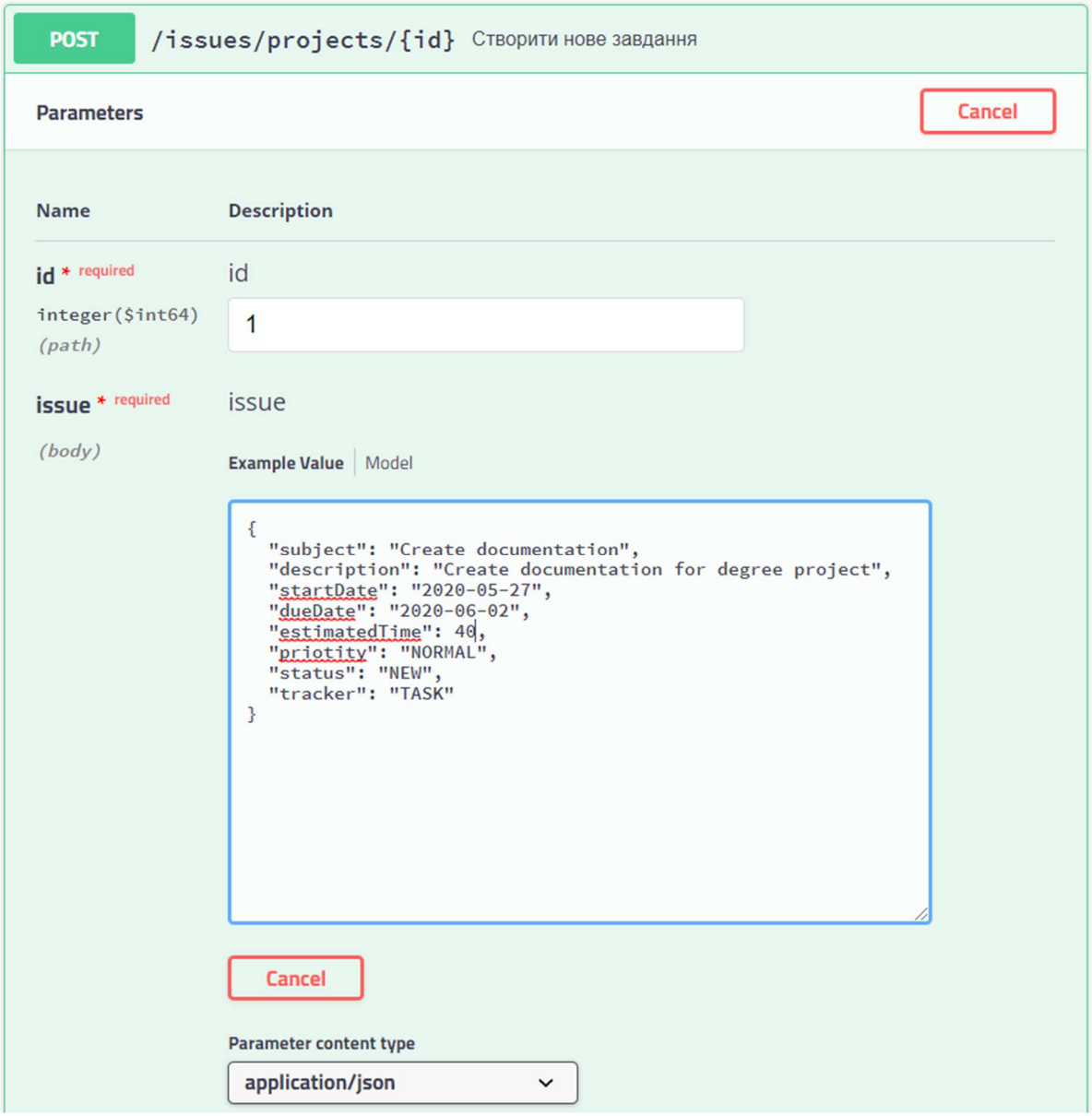


Рис. 3.9. Запит на створення нової задачі.

Рис. 3.10. Запит на присвоєння задачі до користувача.

Переконаемося, що завдання назначилося на користувача (рис 3.11)

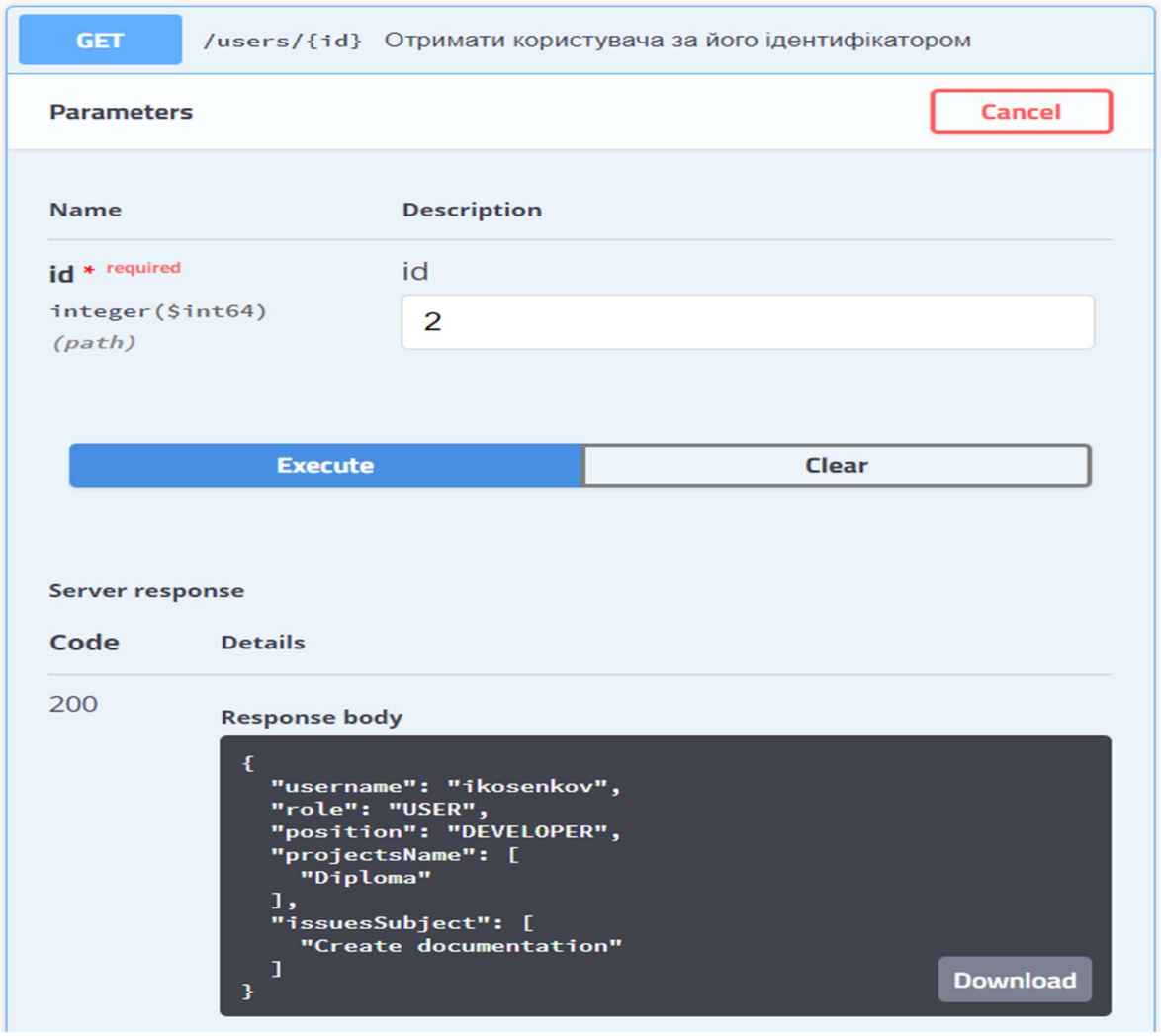


Рис. 3.11. Запит на отримання користувача.

Після того як задача виконана, треба записати в неї час який ми витратили, оновити статус (рис. 3.12), та передати на огляд виконаної роботу відповідальному співробітнику (рис. 3.13).

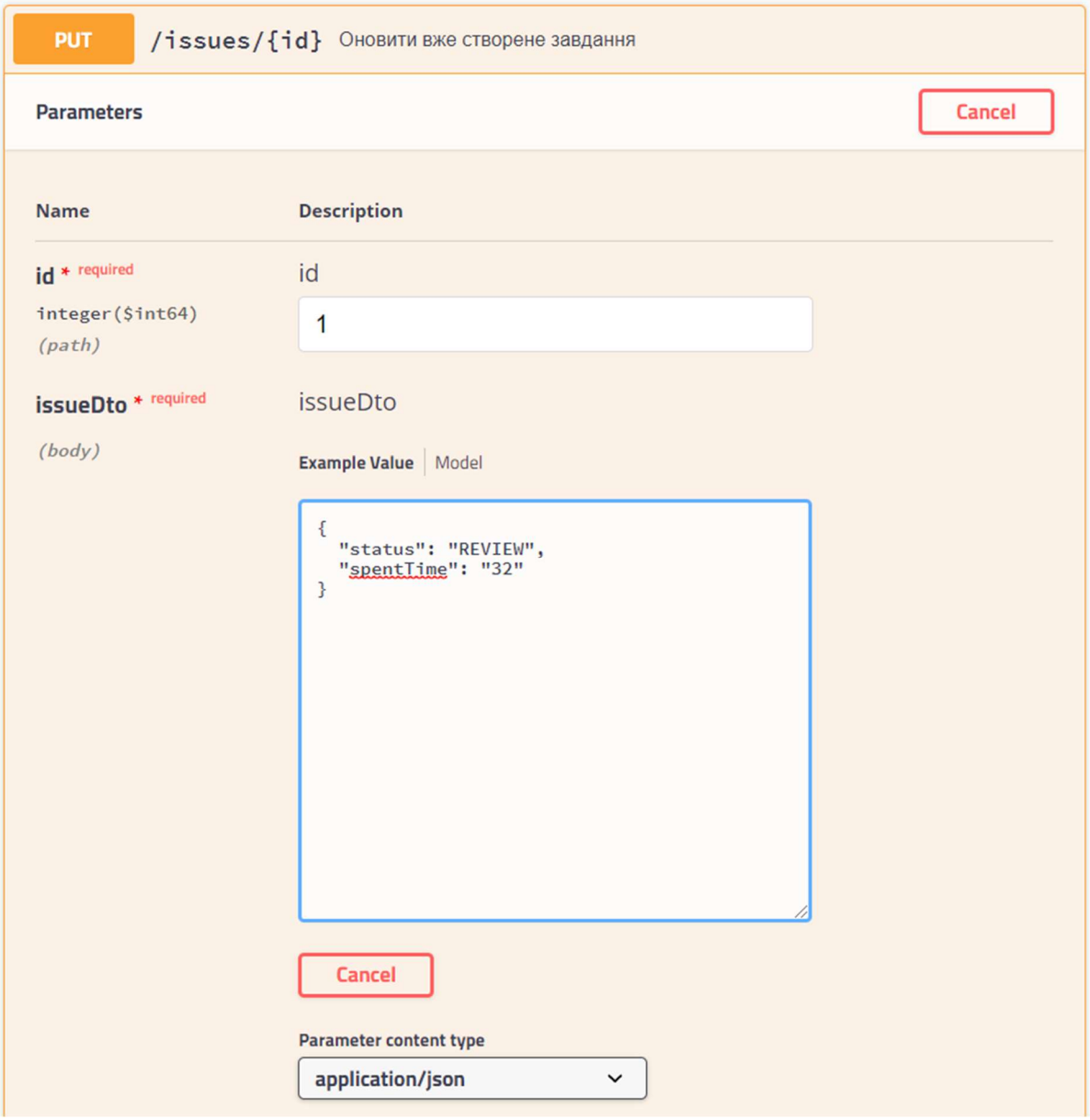


Рис. 3.12. Запит на зміну статусу та використаний час.

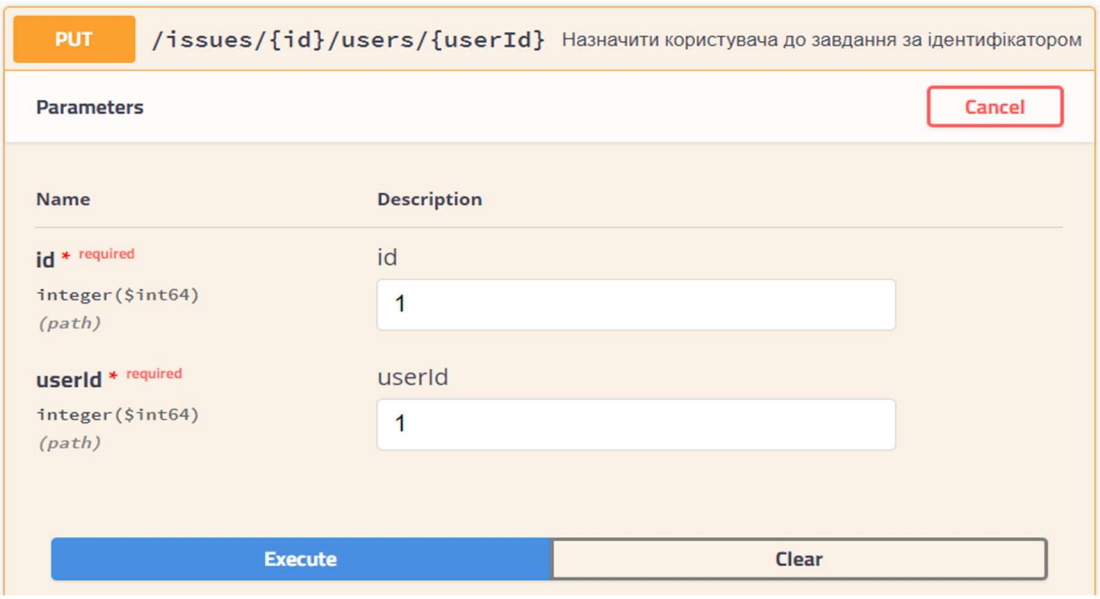


Рис. 3.13. Запит на переназначення завдання на іншого користувача.

Перевіряємо, що задача вже назначена на іншого користувача (рис 3.14).

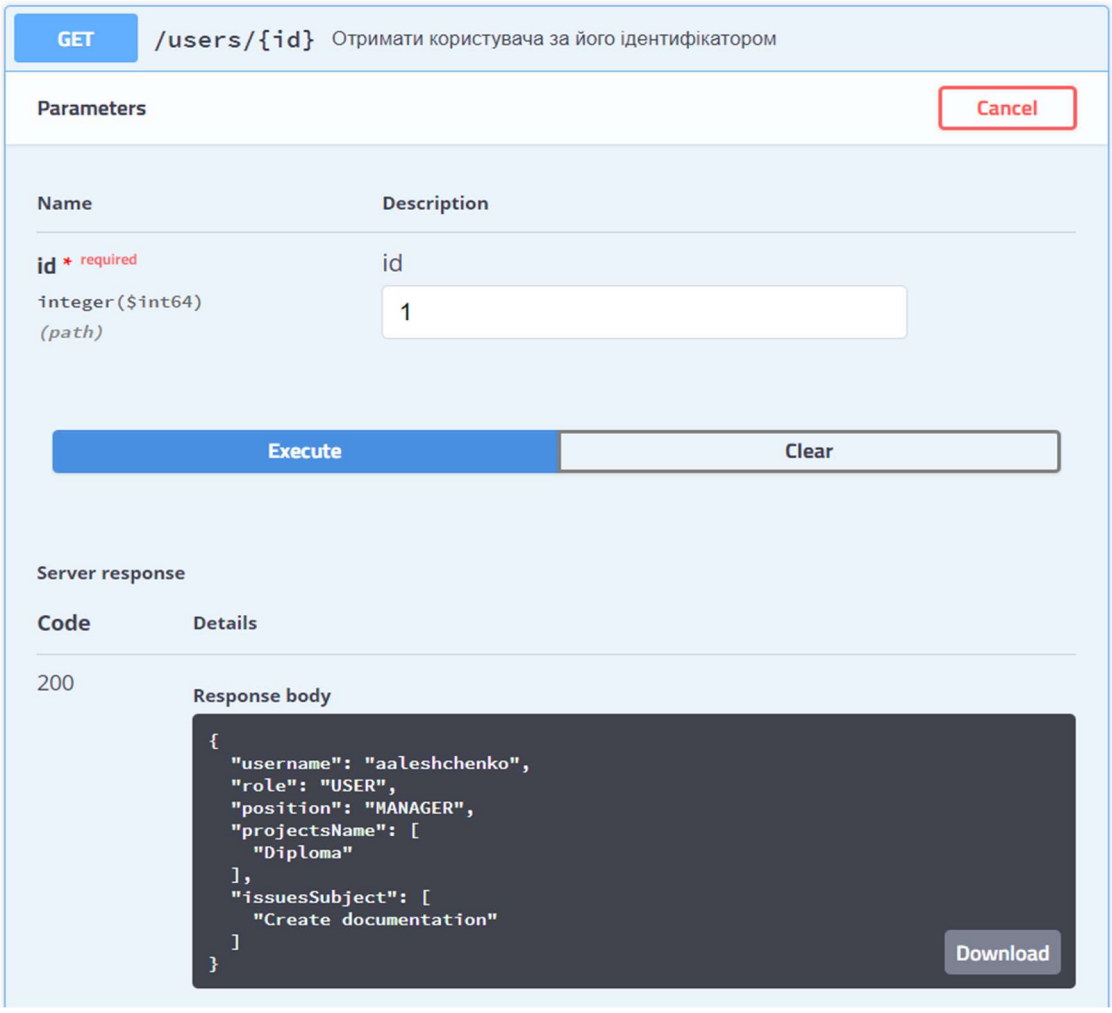


Рис. 3.14. Запит на отримання інформації про користувача.

Далі залишається дописати час, який використав другий співробітник і перевести статус задачі у «вирішена». Вище ми вже робили ці дії, тому покажемо тільки результат виконання програми (рис. 3.15).

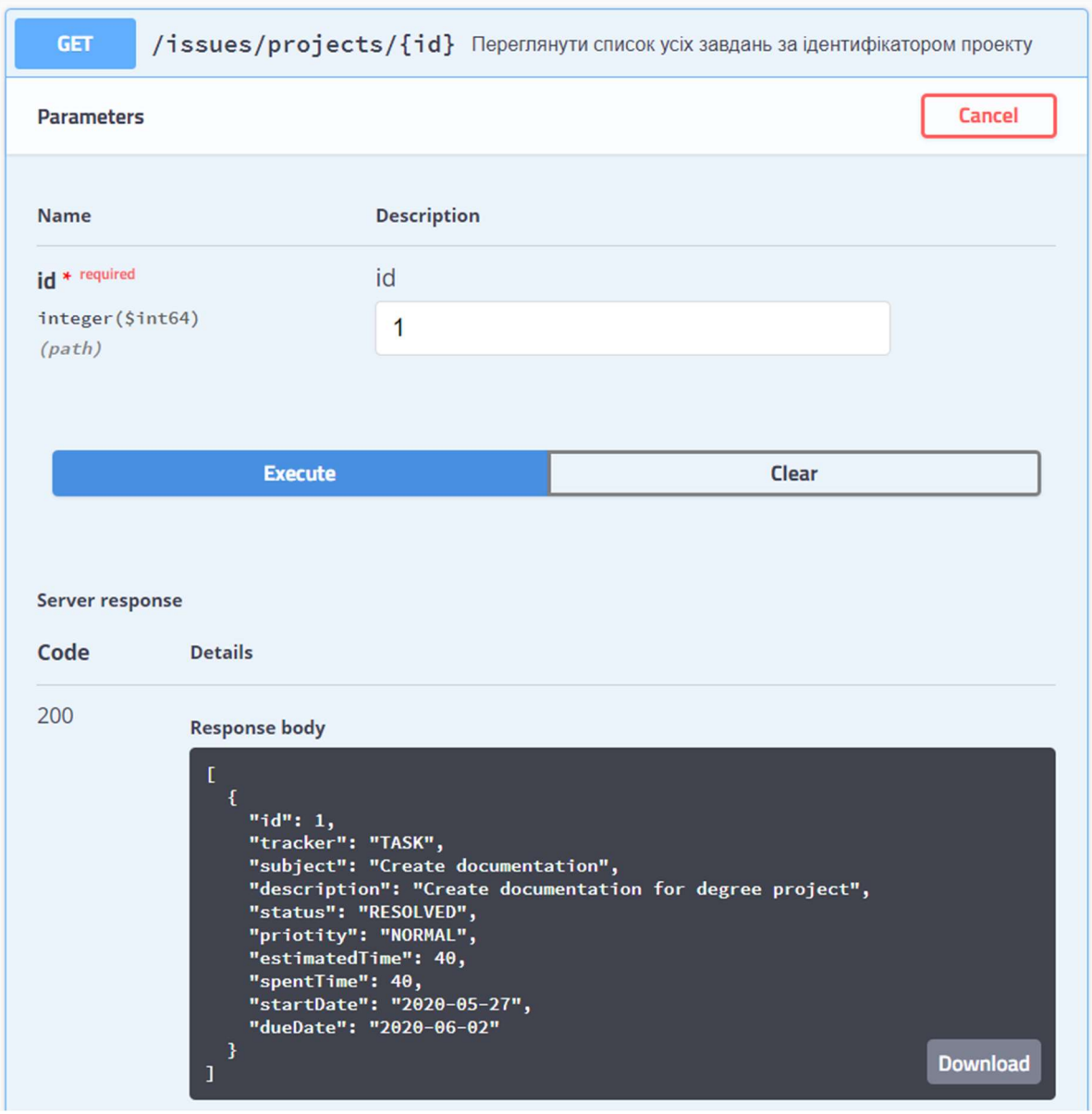


Рис. 3.15. Запит на отримання інформації про завдання.

ВИСНОВКИ ПО РОЗДІЛУ 3

У 3 розділі ми розглянули, що потрібно зробити для запуску програми. Відтворили базовий приклад, як можна користуватися системою. Дані, які були отримані на основі відповідей серверу, можна обробляти і показувати змістовну інформацію по задачам у проекті, на конкретному користувачі тощо.

Також одразу була показана документація на проект, за допомогою якої можна зручно подивитися які кінцеві точки за що відповідають, які дані вони приймають, а які повертають, це найважливіші аспекти для роботи з прикладним програмним інтерфейсом.

ВИСНОВОК

Результатом дипломної роботи є прикладний програмний інтерфейс для системи управління проектами. За допомогою нього можна використовувати будь-яке відображення системи. Наприклад: веб-інтерфейс, мобільний додаток, чат-бот тощо.

Розроблений прикладний програмний інтерфейс має такі можливості як створення проекту; створення завдань, прив’язаних до проекту; створення користувачів; додавання користувачів до проекту; призначення користувачів до завданяя; запис часу, який було витрачено на завданння.

На даному етапі розробки не вдалося зробити більш інформативне відображення часу який був витрачений на завдання.

Цей програмний продукт є початком створення повноцінної системи розподілу завдань між розробниками. Якщо підключити до розробки спеціалістів з інших галузей інформаційних технологій, то можна буде розширити функціонал до веб-сайту, мобільного додатку та чат-боту, щоб люди на одному проекті могли обирати канал зв’язку із системою.

При подальшому інвестуванні часу та коштів, ця система зможе використовуватись у проектах з різних галузей і різних маштабів. Вона буде конкурувати з вже готовими рішеннями які є на ринку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Организация вашей жизни с помощью Trello [Електронний ресурс] – Режим доступу до ресурсу: http://aphd.ua/orhanyzatsyia-vashei-zhyzny-s-pomoshchiu-trello/.
2. Что такое Trello и как им пользоваться [Електронний ресурс] – Режим доступу до ресурсу: https://netology.ru/blog/trello.
3. Jira [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Jira.
4. The #1 software development tool used by agile teams [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/jira.
5. Обзор лучших систем управления проектами [Електронний ресурс] – Режим доступу до ресурсу: https://www.fewskills.com/workflow-project-app/.
6. Какую систему для управления проектами выбрать? 5 лучших таск менеджеров для IT компаний. [Електронний ресурс] – Режим доступу до ресурсу: https://venyoo.ru/blog/kakuyu-sistemu-dlya-upravleniya-proektami-vybrat/.
7. The Pros and Cons of Using Jira Software [Електронний ресурс] – Режим доступу до ресурсу: https://project-management.com/the-pros-and-cons-of-using-jira-software/.
8. Trello vs. Jira for agile project management [Електронний ресурс] – Режим доступу до ресурсу: https://deviniti.com/atlassian/trello-vs-jira-for-agile-project-management/.
9. Redmine - система управления проектами [Електронний ресурс] – Режим доступу до ресурсу: https://www.dvbi.ru/articles/reading/Redmine-tasks-management-system.
10. How does Redmine compare to JIRA and what are the pros and cons for and against each? [Електронний ресурс] – Режим доступу до ресурсу: https://www.quora.com/How-does-Redmine-compare-to-JIRA-and-what-are-the-pros-and-cons-for-and-against-each.

11. Jira vs Redmine [Електронний ресурс] – Режим доступу до ресурсу: https://www.educba.com/jira-vs-redmine/.

12. Asana [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Asana.

13. What is Asana? Task management tracking made easy [Електронний ресурс] – Режим доступу до ресурсу: https://www.computerworld.com/article/3235710/what-is-asana-task-management-tracking-made-easy.html.

14.

15. REST [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/REST.

16. What Is Java? A Beginner's Guide to Java and Its Evolution [Електронний ресурс] – Режим доступу до ресурсу: https://www.edureka.co/blog/what-is-java/.

17. Introduction to JVM, JDK, JRE [Електронний ресурс] – Режим доступу до ресурсу: https://www.techbeamers.com/introduction-jvm-jdk-jre/.

18. Супрун А. Справочник по Java Collections Framework [Електронний ресурс] / Алексей Супрун – Режим доступу до ресурсу: https://habr.com/ru/post/237043/.

19. Java - Collections Framework [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/java/java_collections.htm.

20. Introduction to Java Collections Framework [Електронний ресурс] – Режим доступу до ресурсу: https://www.startertutorials.com/corejava/introduction-java-collections-framework.html.

21. Štechmüller P. Java Collections Framework [Електронний ресурс] / Petr Štechmüller – Режим доступу до ресурсу: https://www.ict.social/java/collections-and-streams-in-java/java-collections-framework.

22. Štechmüller P. Lists with arrays in Java [Електронний ресурс] / Petr Štechmüller – Режим доступу до ресурсу: https://www.ict.social/java/collections-and-streams-in-java/lists-with-arrays-in-java.

23. Štechmüller P. Linked Lists in Java [Електронний ресурс] / Petr Štechmüller – Режим доступу до ресурсу: https://www.ict.social/java/collections-and-streams-in-java/linked-lists-in-java.

24. Jenkov J. Java Set [Електронний ресурс] / Jakob Jenkov – Режим доступу до ресурсу: http://tutorials.jenkov.com/java-collections/set.html.

25. Java Queue – Queue in Java [Електронний ресурс] – Режим доступу до ресурсу: https://www.journaldev.com/13244/java-queue.

26. Spring Framework Overview [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html.

27. Introduction to Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/introduction-to-spring-framework/.

ДОДАТОК 1

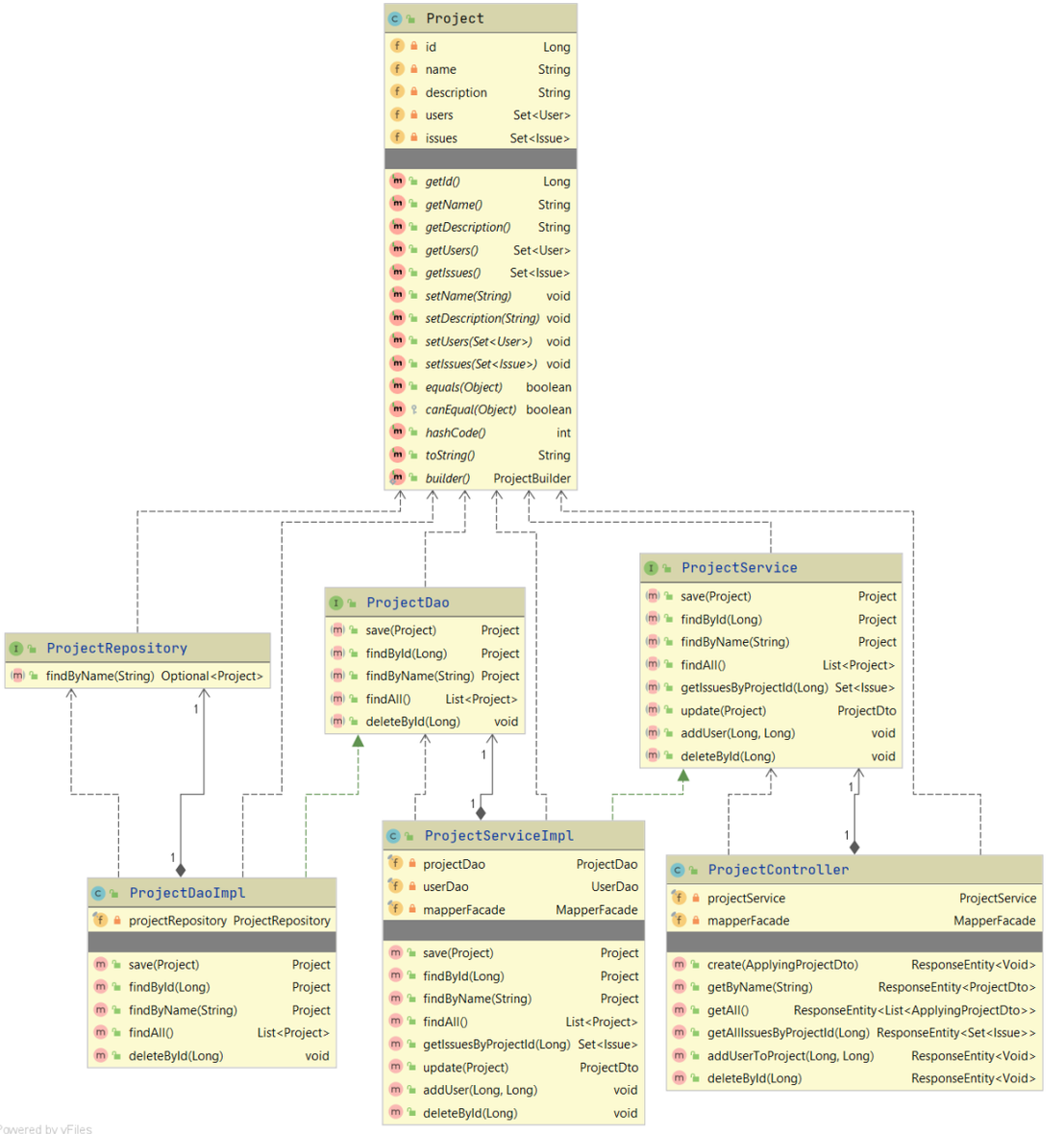
Система розподілу завдань між розробниками

Схема структурна - діаграма класів

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2020 р.



					ІАЛЦ.467100.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Косенков І.В.			Схема структурна Діаграма класів	Літ. / Маса / Масштаб
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			Дипломна робота	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						

ДОДАТОК 2

Система розподілу завдань між розробниками

Схема функціональна - блок-схема алгоритму

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2020 р.

					ІАЛЦ.467100.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	*Схема функціональна Блок-схема алгоритму*	Літ. / Маса / Масштаб
Розроб.		Косенков І.В.				
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			*Дипломна робота*	*КПІ ФІОТ кафедра ОТ гр. ІО-63*
Затверд.						

ДОДАТОК 3

Система розподілу завдань між розробниками

Схема взаємодії

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ 2020 р.

					ІАЛЦ.467100.006 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата	*Схема взаємодії*	Літ. / Маса / Масштаб
Розроб.		Косенков І.В.				
Перевір.		Алещенко О.В.				Арк. / Аркушів
Н. контр.		Сімоненко В.П.			*Дипломна робота*	КПІ ФІОТ кафедра ОТ гр. ІО-63
Затверд.						